Programming Assignment 3: Collinear Points | collinear.zip

Help Center

Submission	
Submission time	Thu-24-Sep 14:25:54
Raw Score	100.00 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  41/41 tests passed
Memory:       1/1 tests passed
Timing:       41/41 tests passed

Aggregate score: 100.00% [Correctness: 65%, Memory: 10%, Timing: 25
%, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 28K
-rw-r--r-- 1 2.9K Sep 24 21:26 BruteCollinearPoints.java
-rw-r--r-- 1 4.3K Sep 24 21:26 FastCollinearPoints.java
-rw-r--r-- 1 4.5K Sep 24 21:26 Point.java
-rw-r--r-- 1 4.1K Sep 24 21:26 studentSubmission.zip


*******************************************************************
***********
```

```
*  compiling
*****************************************************************
***********


% javac Point.java
*-----------------------------------------------------------


================================================================

% javac BruteCollinearPoints.java
*-----------------------------------------------------------


================================================================

% javac FastCollinearPoints.java
*-----------------------------------------------------------


================================================================



% checkstyle *.java
*-----------------------------------------------------------
================================================================



% findbugs *.class
*-----------------------------------------------------------
================================================================



Testing the APIs of your programs.
*-----------------------------------------------------------
Point:

BruteCollinearPoints:

FastCollinearPoints:

================================================================



*****************************************************************
```

```
***********
*  correctness
**********************************************************************
***********

Testing methods in Point
*-----------------------------------------------------------
Running 3 total tests.

Test 1: p.slopeTo(q)
  *  positive infinite slope, where p and q have coordinates in [0,
500)
  *  positive infinite slope, where p and q have coordinates in [0,
32768)
  *  negative infinite slope, where p and q have coordinates in [0,
500)
  *  negative infinite slope, where p and q have coordinates in [0,
32768)
  *  positive zero     slope, where p and q have coordinates in [0,
500)
  *  positive zero     slope, where p and q have coordinates in [0,
32768)
  *  symmetric  for random points p and q with coordinates in [0, 5
00)
  *  symmetric  for random points p and q with coordinates in [0, 3
2768)
  *  transitive for random points p, q, and r with coordinates in [
0, 500)
  *  transitive for random points p, q, and r with coordinates in [
0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 500)
  *  slopeTo(), where p and q have coordinates in [0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 10)
  *  throw a java.lang.NullPointerException if argument is null
==> passed

Test 2: p.compareTo(q)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p and q have coordinates in [0, 500)
  *  antisymmetric, where p and q have coordinates in [0, 32768)
  *  transitive, where p, q, and r have coordinates in [0, 500)
  *  transitive, where p, q, and r have coordinates in [0, 32768)
  *  sign of compareTo(), where p and q have coordinates in [0, 500
)
  *  sign of compareTo(), where p and q have coordinates in [0, 327
68)
  *  sign of compareTo(), where p and q have coordinates in [0, 10)
  *  throw java.lang.NullPointerException exception if argument is
null
==> passed

Test 3: p.slopeOrder().compare(q, r)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p, q, and r have coordinates in [0, 500)
  *  antisymmetric, where p, q, and r have coordinates in [0, 32768
)
  *  transitive, where p, q, r, and s have coordinates in [0, 500)
  *  transitive, where p, q, r, and s have coordinates in [0, 32768
)
  *  sign of compare(), where p, q, and r have coordinates in [0, 5
00)
  *  sign of compare(), where p, q, and r have coordinates in [0, 3
2768)
  *  sign of compare(), where p, q, and r have coordinates in [0, 1
0)
  *  throw java.lang.NullPointerException if either argument is nul
l
==> passed


Total: 3/3 tests passed!

================================================================

********************************************************************
***********
*  correctness (using reference Point.java and LineSegment.java)
********************************************************************
***********

Testing methods in BruteCollinearPoints
*------------------------------------------------------------
Running 17 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - no 5 (or more) points are collinear
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
==> passed

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
```

```
==> passed

Test 5: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 6: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 7: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 8: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 9: Random line segments
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 10: Check that data type is immutable by testing whether each
method
        returns the same value, regardless of any intervening opera
tions
  *  input8.txt
  *  equidistant.txt
==> passed
```

```
Test 11: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
  *  equidistant.txt
==> passed

Test 12: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 13: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 14: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 17/17 tests passed!

================================================================

Testing methods in FastCollinearPoints
*------------------------------------------------------------
Running 21 total tests.
```

```
The inputs satisfy the following conditions:
  - no duplicate points
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> passed

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
  *  filename = horizontal50.txt
  *  filename = horizontal75.txt
  *  filename = horizontal100.txt
==> passed

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
  *  filename = vertical50.txt
  *  filename = vertical75.txt
  *  filename = vertical100.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
```

```
  *  filename = random23.txt
  *  filename = random38.txt
  *  filename = random91.txt
  *  filename = random152.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5a: Points from a file with 5 or more on some line segments
  *  filename = input9.txt
  *  filename = input10.txt
  *  filename = input20.txt
  *  filename = input50.txt
  *  filename = input80.txt
  *  filename = input300.txt
  *  filename = inarow.txt
==> passed

Test 5b: Points from a file with 5 or more on some line segments
  *  filename = kw1260.txt
  *  filename = rs1423.txt
==> passed

Test 6: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 7: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> passed

Test 8: Check for fragile dependence on return value of toString()
```

```
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 9: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  25 random line segments
  *  50 random line segments
  *  100 random line segments
==> passed

Test 10: Random line segments
  *   1 random line segment
  *   5 random line segments
  *  25 random line segments
  *  50 random line segments
  *  100 random line segments
==> passed

Test 11: Random distinct points in a given range
  *  5 random points in a 10-by-10 grid
  *  10 random points in a 10-by-10 grid
  *  50 random points in a 10-by-10 grid
  *  90 random points in a 10-by-10 grid
  *  200 random points in a 50-by-50 grid
==> passed

Test 12: M*N points on an M-by-N grid
  *  3-by-3 grid
  *  4-by-4 grid
  *  5-by-5 grid
  *  10-by-10 grid
  *  20-by-20 grid
  *  5-by-4 grid
  *  6-by-4 grid
  *  10-by-4 grid
  *  15-by-4 grid
  *  25-by-4 grid
==> passed

Test 13: Check that data type is immutable by testing whether each
method
```

```
         returns the same value, regardless of any intervening oper
ations
  *  input8.txt
  *  equidistant.txt
==> passed

Test 14: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
  *  equidistant.txt
==> passed

Test 15: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 16: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 17: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 21/21 tests passed!
```

```
================================================================

*****************************************************************
***********
*  memory
*****************************************************************
***********

Computing memory of Point
*-----------------------------------------------------------
Running 1 total tests.

The maximum amount of memory per Point object is 32 bytes.

Student memory = 24 bytes (passed)

Total: 1/1 tests passed!

================================================================



*****************************************************************
***********
*  timing
*****************************************************************
***********

Timing BruteCollinearPoints
*-----------------------------------------------------------
Running 10 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
-----------------------------------------------------------------
----------------------------
=> passed    16   0.02        680          0           680
44
=> passed    32   0.00       5456          0          5456
```

```
124
=> passed    64   0.00       43680           0          43680
304
=> passed   128   0.01      349504           0         349504
746
=> passed   256   0.05     2796160           0        2796160
1742
==> 5/5 tests passed

Test 2a-2e: Find collinear points among N/4 arbitrary line segments



                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
-----------------------------------------------------------------------
----------------------------
=> passed    16   0.00         766           0            766
44
=> passed    32   0.00        5828           0           5828
122
=> passed    64   0.00       45348           0          45348
304
=> passed   128   0.01      356418           0         356418
735
=> passed   256   0.04     2822979           0        2822979
1734
==> 5/5 tests passed

Total: 10/10 tests passed!

=================================================================




Timing FastCollinearPoints
*-------------------------------------------------------------
Running 31 total tests.

Test 1a-1e: Find collinear points among N random distinct points



                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
```

```
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.03       12095      18306          48707
373
=> passed   128   0.02       48767      87962         224691
868
=> passed   256   0.04      195839     411496        1018831
1992
=> passed   512   0.27      784895    1877597        4540089
4507
=> passed  1024   0.58     3142637    8505039       20152715
9998
=> passed  2048   1.82    12576607   37788638       88153883
22047
==> 6/6 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (88153883 / 20152715) = 2.13
=> passed

==> 7/7 tests passed

Test 2a-2e: Find collinear points among the N points on an N-by-1 g
rid

                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        4159       3968          12095
432
=> passed   128   0.00       16511      16128          48767
1000
=> passed   256   0.01       65791      65024         195839
2249
=> passed   512   0.02      262655     261120         784895
4983
=> passed  1024   0.06     1049599    1046528        3142655
11014
=> passed  2048   0.19     4196351    4190208       12576767
24091
=> passed  4096   0.39    16781311   16769024       50319359
52223
```

```
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (50319359 / 12576767) = 2.00
=> passed

==> 8/8 tests passed

Test 3a-3e: Find collinear points among the 4N points on an N/4-by-
4 grid

                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
---------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8423       16222          40867
774
=> passed   128   0.00       33575       58163         149901
2356
=> passed   256   0.02      134055      152338         438731
7711
=> passed   512   0.03      535719      546156        1628031
26828
=> passed  1024   0.10     2141863     2080221        6302305
98397
=> passed  2048   0.34     8565415     8125112       24815639
373629
=> passed  4096   1.34    34257575    32104200       98465975
1450353
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (98465975 / 24815639) = 1.99
=> passed

==> 8/8 tests passed

Test 4a-4e: Find collinear points among the 8N points on an N/8-by-
8 grid

                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
---------------------------------------------------------------------
----------------------------
```

```
=> passed    64   0.00        8471       17691           43853
763
=> passed   128   0.00       33823       80299          194421
2267
=> passed   256   0.01      135087      311802          758691
7375
=> passed   512   0.04      539919      859701         2259321
25473
=> passed  1024   0.14     2158695     3252409         8663513
92922
=> passed  2048   0.47     8632807    12677306        33987419
351757
=> passed  4096   1.91    34527191    50035440       134598071
1362957
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (134598071 / 33987419) = 1.9
9
=> passed

==> 8/8 tests passed

Total: 31/31 tests passed!

================================================================
```

Submission	
Submission time	Thu-24-Sep 14:21:13
Raw Score	100.00 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        FAILED
Findbugs:     No potential bugs found.
```

```
API:          PASSED

Correctness:  41/41 tests passed
Memory:       1/1 tests passed
Timing:       41/41 tests passed

Aggregate score: 100.00% [Correctness: 65%, Memory: 10%, Timing: 25
%, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 28K
-rw-r--r-- 1 2.9K Sep 24 21:21 BruteCollinearPoints.java
-rw-r--r-- 1 4.3K Sep 24 21:21 FastCollinearPoints.java
-rw-r--r-- 1 4.5K Sep 24 21:21 Point.java
-rw-r--r-- 1 4.1K Sep 24 21:21 studentSubmission.zip


*******************************************************************
***********
*  compiling
*******************************************************************
***********


% javac Point.java
*-----------------------------------------------------------


================================================================


% javac BruteCollinearPoints.java
*-----------------------------------------------------------


================================================================


% javac FastCollinearPoints.java
*-----------------------------------------------------------


================================================================
% checkstyle *.java
*-----------------------------------------------------------
Point.java:131:34: ',' is not followed by whitespace.
Point.java:133:30: ',' is not followed by whitespace.
Point.java:134:30: ',' is not followed by whitespace.
Point.java:135:58: ',' is not followed by whitespace.
FastCollinearPoints.java:3:8: Unused import statement for 'edu.prin
ceton.cs.algs4.StdDraw'.
Checkstyle ends with 5 errors.
================================================================


% findbugs *.class
*-----------------------------------------------------------
================================================================


Testing the APIs of your programs.
*-----------------------------------------------------------
Point:

BruteCollinearPoints:

FastCollinearPoints:

================================================================


*******************************************************************
***********
*  correctness
*******************************************************************
***********

Testing methods in Point
*-----------------------------------------------------------
Running 3 total tests.

Test 1: p.slopeTo(q)
  *  positive infinite slope, where p and q have coordinates in [0,
500)
  *  positive infinite slope, where p and q have coordinates in [0,
32768)
```

```
  *  negative infinite slope, where p and q have coordinates in [0, 
500)
  *  negative infinite slope, where p and q have coordinates in [0, 
32768)
  *  positive zero     slope, where p and q have coordinates in [0, 
500)
  *  positive zero     slope, where p and q have coordinates in [0, 
32768)
  *  symmetric  for random points p and q with coordinates in [0, 5
00)
  *  symmetric  for random points p and q with coordinates in [0, 3
2768)
  *  transitive for random points p, q, and r with coordinates in [
0, 500)
  *  transitive for random points p, q, and r with coordinates in [
0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 500)
  *  slopeTo(), where p and q have coordinates in [0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 10)
  *  throw a java.lang.NullPointerException if argument is null
==> passed

Test 2: p.compareTo(q)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p and q have coordinates in [0, 500)
  *  antisymmetric, where p and q have coordinates in [0, 32768)
  *  transitive, where p, q, and r have coordinates in [0, 500)
  *  transitive, where p, q, and r have coordinates in [0, 32768)
  *  sign of compareTo(), where p and q have coordinates in [0, 500
)
  *  sign of compareTo(), where p and q have coordinates in [0, 327
68)
  *  sign of compareTo(), where p and q have coordinates in [0, 10)
  *  throw java.lang.NullPointerException exception if argument is 
null
==> passed

Test 3: p.slopeOrder().compare(q, r)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p, q, and r have coordinates in [0, 500)
  *  antisymmetric, where p, q, and r have coordinates in [0, 32768
)
```

```
  *  transitive, where p, q, r, and s have coordinates in [0, 500)
  *  transitive, where p, q, r, and s have coordinates in [0, 32768
)
  *  sign of compare(), where p, q, and r have coordinates in [0, 5
00)
  *  sign of compare(), where p, q, and r have coordinates in [0, 3
2768)
  *  sign of compare(), where p, q, and r have coordinates in [0, 1
0)
  *  throw java.lang.NullPointerException if either argument is nul
l
==> passed


Total: 3/3 tests passed!

================================================================

******************************************************************
***********
*  correctness (using reference Point.java and LineSegment.java)
******************************************************************
***********

Testing methods in BruteCollinearPoints
*-----------------------------------------------------------
Running 17 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - no 5 (or more) points are collinear
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
==> passed
```

```
Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 6: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
```

```
==> passed

Test 7: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 8: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 9: Random line segments
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 10: Check that data type is immutable by testing whether each
method
        returns the same value, regardless of any intervening opera
tions
  *  input8.txt
  *  equidistant.txt
==> passed

Test 11: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
  *  equidistant.txt
==> passed

Test 12: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
```

```
==> passed

Test 13: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 14: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 17/17 tests passed!

================================================================

Testing methods in FastCollinearPoints
*-----------------------------------------------------------
Running 21 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> passed

Test 2a: Points from a file with horizontal line segments
```

```
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
  *  filename = horizontal50.txt
  *  filename = horizontal75.txt
  *  filename = horizontal100.txt
==> passed

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
  *  filename = vertical50.txt
  *  filename = vertical75.txt
  *  filename = vertical100.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
  *  filename = random91.txt
  *  filename = random152.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5a: Points from a file with 5 or more on some line segments
```

```
  *  filename = input9.txt
  *  filename = input10.txt
  *  filename = input20.txt
  *  filename = input50.txt
  *  filename = input80.txt
  *  filename = input300.txt
  *  filename = inarow.txt
==> passed

Test 5b: Points from a file with 5 or more on some line segments
  *  filename = kw1260.txt
  *  filename = rs1423.txt
==> passed

Test 6: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 7: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> passed

Test 8: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 9: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  25 random line segments
  *  50 random line segments
  *  100 random line segments
==> passed

Test 10: Random line segments
```

```
  *   1 random line segment
  *   5 random line segments
  *  25 random line segments
  *  50 random line segments
  *  100 random line segments
==> passed

Test 11: Random distinct points in a given range
  *  5 random points in a 10-by-10 grid
  *  10 random points in a 10-by-10 grid
  *  50 random points in a 10-by-10 grid
  *  90 random points in a 10-by-10 grid
  *  200 random points in a 50-by-50 grid
==> passed

Test 12: M*N points on an M-by-N grid
  *  3-by-3 grid
  *  4-by-4 grid
  *  5-by-5 grid
  *  10-by-10 grid
  *  20-by-20 grid
  *  5-by-4 grid
  *  6-by-4 grid
  *  10-by-4 grid
  *  15-by-4 grid
  *  25-by-4 grid
==> passed

Test 13: Check that data type is immutable by testing whether each
method
         returns the same value, regardless of any intervening oper
ations
  *  input8.txt
  *  equidistant.txt
==> passed

Test 14: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
  *  equidistant.txt
==> passed

Test 15: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 16: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 17: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 21/21 tests passed!

================================================================

*******************************************************************
***********
*  memory
*******************************************************************
***********

Computing memory of Point
*-----------------------------------------------------------
Running 1 total tests.

The maximum amount of memory per Point object is 32 bytes.
```

```
Student memory = 24 bytes (passed)

Total: 1/1 tests passed!

================================================================



********************************************************************
***********
*  timing
********************************************************************
***********

Timing BruteCollinearPoints
*------------------------------------------------------------
Running 10 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    16   0.02         680           0            680
48
=> passed    32   0.00        5456           0           5456
125
=> passed    64   0.00       43680           0          43680
300
=> passed   128   0.01      349504           0         349504
742
=> passed   256   0.05     2796160           0        2796160
1736
==> 5/5 tests passed

Test 2a-2e: Find collinear points among N/4 arbitrary line segments


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
```

```
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    16   0.00         770           0           770
47
=> passed    32   0.00        5850           0          5850
124
=> passed    64   0.00       45364           0         45364
306
=> passed   128   0.01      355998           0        355998
736
=> passed   256   0.05     2821638           0       2821638
1715
==> 5/5 tests passed

Total: 10/10 tests passed!

=================================================================




Timing FastCollinearPoints
*-------------------------------------------------------------
Running 31 total tests.

Test 1a-1e: Find collinear points among N random distinct points



                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.03       12095       18473         49041
370
=> passed   128   0.02       48767       87412        223591
860
=> passed   256   0.06      195839      409342       1014523
1987
=> passed   512   0.21      784895     1875318       4535531
4516
=> passed  1024   0.61     3142631     8477913      20098457
9968
=> passed  2048   1.87    12576647    37860557      88297761
```

```
22005
==> 6/6 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (88297761 / 20098457) = 2.14
=> passed

==> 7/7 tests passed

Test 2a-2e: Find collinear points among the N points on an N-by-1 g
rid

                                                          slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        4159        3968          12095
437
=> passed   128   0.00       16511       16128          48767
993
=> passed   256   0.01       65791       65024         195839
2248
=> passed   512   0.03      262655      261120         784895
4996
=> passed  1024   0.06     1049599     1046528        3142655
11004
=> passed  2048   0.17     4196351     4190208       12576767
24060
=> passed  4096   0.35    16781311    16769024       50319359
52228
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (50319359 / 12576767) = 2.00
=> passed

==> 8/8 tests passed

Test 3a-3e: Find collinear points among the 4N points on an N/4-by-
4 grid

                                                          slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
```

```
----------------------------
=> passed    64   0.00        8423       16222          40867
777
=> passed   128   0.01       33575       58163         149901
2362
=> passed   256   0.03      134055      152338         438731
7714
=> passed   512   0.08      535719      546156        1628031
26854
=> passed  1024   0.17     2141863     2080221        6302305
98378
=> passed  2048   0.34     8565415     8125112       24815639
373591
=> passed  4096   1.32    34257575    32104200       98465975
1450369
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (98465975 / 24815639) = 1.99
=> passed

==> 8/8 tests passed

Test 4a-4e: Find collinear points among the 8N points on an N/8-by-
8 grid

                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8471       17691          43853
765
=> passed   128   0.00       33823       80299         194421
2289
=> passed   256   0.01      135087      311802         758691
7373
=> passed   512   0.05      539919      859701        2259321
25479
=> passed  1024   0.15     2158695     3252409        8663513
92934
=> passed  2048   0.46     8632807    12677306       33987419
351758
=> passed  4096   1.86    34527191    50035440      134598071
1362953
```

```
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (134598071 / 33987419) = 1.9
9
=> passed

==> 8/8 tests passed

Total: 31/31 tests passed!

================================================================
```

Submission	
Submission time	Thu-24-Sep 14:19:27
Raw Score	0.00 / 100.00

Feedback	`Compilation:  FAILED`
	
	`BruteCollinearPoints.java failed to compile, javac reports:`
	
	`BruteCollinearPoints.java could not be found.`

Submission	
Submission time	Thu-24-Sep 14:00:21
Raw Score	77.80 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        FAILED
Findbugs:     No potential bugs found.
```

```
API:          PASSED

Correctness:  27/41 tests passed
Memory:       1/1 tests passed
Timing:       41/41 tests passed

Aggregate score: 77.80% [Correctness: 65%, Memory: 10%, Timing: 25%
, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 24K
-rw-r--r-- 1 2.9K Sep 24 21:01 BruteCollinearPoints.java
-rw-r--r-- 1 4.1K Sep 24 21:01 FastCollinearPoints.java
-rw-r--r-- 1 4.5K Sep 24 21:01 Point.java
-rw-r--r-- 1 4.0K Sep 24 21:01 studentSubmission.zip


*******************************************************************
***********
*  compiling
*******************************************************************
***********


% javac Point.java
*-----------------------------------------------------------


================================================================


% javac BruteCollinearPoints.java
*-----------------------------------------------------------


================================================================


% javac FastCollinearPoints.java
*-----------------------------------------------------------


================================================================
% checkstyle *.java
*-----------------------------------------------------------
Point.java:131:34: ',' is not followed by whitespace.
Point.java:133:30: ',' is not followed by whitespace.
Point.java:134:30: ',' is not followed by whitespace.
Point.java:135:58: ',' is not followed by whitespace.
Checkstyle ends with 4 errors.
================================================================


% findbugs *.class
*-----------------------------------------------------------
================================================================


Testing the APIs of your programs.
*-----------------------------------------------------------
Point:

BruteCollinearPoints:

FastCollinearPoints:

================================================================


******************************************************************
***********
*  correctness
******************************************************************
***********

Testing methods in Point
*-----------------------------------------------------------
Running 3 total tests.

Test 1: p.slopeTo(q)
  *  positive infinite slope, where p and q have coordinates in [0,
500)
  *  positive infinite slope, where p and q have coordinates in [0,
32768)
  *  negative infinite slope, where p and q have coordinates in [0,
500)
```

```
  *  negative infinite slope, where p and q have coordinates in [0,
32768)
  *  positive zero     slope, where p and q have coordinates in [0,
500)
  *  positive zero     slope, where p and q have coordinates in [0,
32768)
  *  symmetric  for random points p and q with coordinates in [0, 5
00)
  *  symmetric  for random points p and q with coordinates in [0, 3
2768)
  *  transitive for random points p, q, and r with coordinates in [
0, 500)
  *  transitive for random points p, q, and r with coordinates in [
0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 500)
  *  slopeTo(), where p and q have coordinates in [0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 10)
  *  throw a java.lang.NullPointerException if argument is null
==> passed

Test 2: p.compareTo(q)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p and q have coordinates in [0, 500)
  *  antisymmetric, where p and q have coordinates in [0, 32768)
  *  transitive, where p, q, and r have coordinates in [0, 500)
  *  transitive, where p, q, and r have coordinates in [0, 32768)
  *  sign of compareTo(), where p and q have coordinates in [0, 500
)
  *  sign of compareTo(), where p and q have coordinates in [0, 327
68)
  *  sign of compareTo(), where p and q have coordinates in [0, 10)
  *  throw java.lang.NullPointerException exception if argument is
null
==> passed

Test 3: p.slopeOrder().compare(q, r)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p, q, and r have coordinates in [0, 500)
  *  antisymmetric, where p, q, and r have coordinates in [0, 32768
)
  *  transitive, where p, q, r, and s have coordinates in [0, 500)
  *  transitive, where p, q, r, and s have coordinates in [0, 32768
)
  *  sign of compare(), where p, q, and r have coordinates in [0, 5
00)
  *  sign of compare(), where p, q, and r have coordinates in [0, 3
2768)
  *  sign of compare(), where p, q, and r have coordinates in [0, 1
0)
  *  throw java.lang.NullPointerException if either argument is nul
l
==> passed


Total: 3/3 tests passed!

================================================================

*******************************************************************
***********
*  correctness (using reference Point.java and LineSegment.java)
*******************************************************************
***********

Testing methods in BruteCollinearPoints
*-----------------------------------------------------------
Running 17 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - no 5 (or more) points are collinear
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
==> passed

Test 2b: Random horizontal line segments
```

```
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 6: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed
```

```
Test 7: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 8: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 9: Random line segments
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 10: Check that data type is immutable by testing whether each 
method
        returns the same value, regardless of any intervening opera
tions
  *  input8.txt
  *  equidistant.txt
==> passed

Test 11: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
  *  equidistant.txt
==> passed

Test 12: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed
```

```
Test 13: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 14: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 17/17 tests passed!

================================================================

Testing methods in FastCollinearPoints
*-----------------------------------------------------------
Running 21 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (3000, 4000) -> (14000, 15000)
     -  reference segment 1: (3000, 4000) -> (6000, 7000) -> (14000
, 15000) -> (20000, 21000)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (3000, 4000) -> (14000,
15000)
     -  1 missing entry in student solution: (3000, 4000) -> (6000,
7000) -> (14000, 15000) -> (20000, 21000)

  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> FAILED

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (2682, 14118) -> (7453, 14118)
     -  reference segment 4: (2682, 14118) -> (5067, 14118) -> (745
3, 14118) -> (7821, 14118)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (2682, 14118) -> (7453,
14118)
     -  1 missing entry in student solution: (2682, 14118) -> (5067
, 14118) -> (7453, 14118) -> (7821, 14118)

  *  filename = horizontal25.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 24: (8784, 20913) -> (16352, 20913)
     -  reference segment 24: (8784, 20913) -> (9880, 20913) -> (16
352, 20913) -> (19666, 20913)
```

```
     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (8784, 20913) -> (16352,
20913)
     -  1 missing entry in student solution: (8784, 20913) -> (9880
, 20913) -> (16352, 20913) -> (19666, 20913)

  *  filename = horizontal50.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 49: (5249, 20754) -> (14800, 20754)
     -  reference segment 49: (5249, 20754) -> (5559, 20754) -> (14
800, 20754) -> (17428, 20754)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (5249, 20754) -> (14800,
20754)
     -  1 missing entry in student solution: (5249, 20754) -> (5559
, 20754) -> (14800, 20754) -> (17428, 20754)

  *  filename = horizontal75.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 74: (1536, 20976) -> (14178, 20976)
     -  reference segment 74: (1536, 20976) -> (6545, 20976) -> (14
178, 20976) -> (14591, 20976)

     -  student   solution has 75 non-null entries
     -  reference solution has 75 non-null entries
     -  1 extra entry in student solution: (1536, 20976) -> (14178,
20976)
     -  1 missing entry in student solution: (1536, 20976) -> (6545
, 20976) -> (14178, 20976) -> (14591, 20976)

  *  filename = horizontal100.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 99: (5835, 20698) -> (16154, 20698)
     -  reference segment 99: (5835, 20698) -> (7673, 20698) -> (16
154, 20698) -> (19642, 20698)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (5835, 20698) -> (16154, 
20698)
     -  1 missing entry in student solution: (5835, 20698) -> (7673
, 20698) -> (16154, 20698) -> (19642, 20698)

==> FAILED

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (7941, 3812) -> (11552, 3812)
     -  reference segment 0: (7941, 3812) -> (10960, 3812) -> (1155
2, 3812) -> (13072, 3812)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (7941, 3812) -> (11552, 
3812)
     -  1 missing entry in student solution: (7941, 3812) -> (10960
, 3812) -> (11552, 3812) -> (13072, 3812)

     -  failed on trial 1 of 500
     4
      7941  3812
     13072  3812
     10960  3812
     11552  3812

  *   5 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (1672, 17859) -> (4958, 17859)
     -  reference segment 4: (1672, 17859) -> (3696, 17859) -> (495
8, 17859) -> (12069, 17859)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (1672, 17859) -> (4958, 
17859)
     -  1 missing entry in student solution: (1672, 17859) -> (3696
, 17859) -> (4958, 17859) -> (12069, 17859)
```

```
      -  failed on trial 1 of 250
      20
      20637  4489
       2203  4489
       6899  9881
       6940  2619
       2671  9761
      10222  9761
       6497  4489
      12475  9881
      18187  2619
       3696 17859
      15529  2619
       1672 17859
      19584  9761
      12649  9881
       4958 17859
      19745  2619
      11178  4489
      12069 17859
       8050  9881
       6491  9761

   *  10 random horizontal line segments
      -  segments() contains a subsegment of a segment in reference
solution
      -  student   segment 9: (4362, 14739) -> (8088, 14739)
      -  reference segment 9: (4362, 14739) -> (6876, 14739) -> (808
8, 14739) -> (8867, 14739)

      -  student   solution has 10 non-null entries
      -  reference solution has 10 non-null entries
      -  1 extra entry in student solution: (4362, 14739) -> (8088,
14739)
      -  1 missing entry in student solution: (4362, 14739) -> (6876
, 14739) -> (8088, 14739) -> (8867, 14739)

      -  failed on trial 1 of 50

   *  15 random horizontal line segments
      -  segments() contains a subsegment of a segment in reference
solution
      -  student   segment 14: (1028, 20841) -> (13437, 20841)
      -  reference segment 14: (1028, 20841) -> (2644, 20841) -> (13
437, 20841) -> (18176, 20841)

      -  student   solution has 15 non-null entries
      -  reference solution has 15 non-null entries
      -  1 extra entry in student solution: (1028, 20841) -> (13437,
20841)
      -  1 missing entry in student solution: (1028, 20841) -> (2644
, 20841) -> (13437, 20841) -> (18176, 20841)

      -  failed on trial 1 of 5

==> FAILED

Test 3a: Points from a file with vertical line segments
   *  filename = vertical5.txt
      -  segments() contains a subsegment of a segment in reference
solution
      -  student   segment 2: (5757, 3426) -> (5757, 16647)
      -  reference segment 1: (5757, 3426) -> (5757, 13581) -> (5757
, 16647) -> (5757, 20856)

      -  student   solution has 5 non-null entries
      -  reference solution has 5 non-null entries
      -  1 extra entry in student solution: (5757, 3426) -> (5757, 1
6647)
      -  1 missing entry in student solution: (5757, 3426) -> (5757,
13581) -> (5757, 16647) -> (5757, 20856)

   *  filename = vertical25.txt
      -  segments() contains a subsegment of a segment in reference
solution
      -  student   segment 22: (13536, 9107) -> (13536, 13165)
      -  reference segment 14: (13536, 9107) -> (13536, 9393) -> (13
536, 13165) -> (13536, 20946)

      -  student   solution has 25 non-null entries
      -  reference solution has 25 non-null entries
      -  1 extra entry in student solution: (13536, 9107) -> (13536,
13165)
      -  1 missing entry in student solution: (13536, 9107) -> (1353
6, 9393) -> (13536, 13165) -> (13536, 20946)

   *  filename = vertical50.txt
      -  segments() contains a subsegment of a segment in reference
```

```
solution
     -  student   segment 5: (10695, 1287) -> (10695, 20756)
     -  reference segment 27: (10695, 1287) -> (10695, 10521) -> (1
0695, 20756) -> (10695, 20927)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (10695, 1287) -> (10695,
20756)
     -  1 missing entry in student solution: (10695, 1287) -> (1069
5, 10521) -> (10695, 20756) -> (10695, 20927)

  *  filename = vertical75.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 45: (18293, 5438) -> (18293, 19756)
     -  reference segment 66: (18293, 5438) -> (18293, 17680) -> (1
8293, 19756) -> (18293, 20983)

     -  student   solution has 75 non-null entries
     -  reference solution has 75 non-null entries
     -  1 extra entry in student solution: (18293, 5438) -> (18293,
19756)
     -  1 missing entry in student solution: (18293, 5438) -> (1829
3, 17680) -> (18293, 19756) -> (18293, 20983)

  *  filename = vertical100.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 84: (19597, 8445) -> (19597, 17520)
     -  reference segment 93: (19597, 8445) -> (19597, 10925) -> (1
9597, 17520) -> (19597, 20918)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (19597, 8445) -> (19597,
17520)
     -  1 missing entry in student solution: (19597, 8445) -> (1959
7, 10925) -> (19597, 17520) -> (19597, 20918)

==> FAILED

Test 3b: Random vertical line segments
  *   1 random vertical line segment
```

```
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (13636, 4463) -> (13636, 13555)
     -  reference segment 0: (13636, 4463) -> (13636, 6279) -> (136
36, 13555) -> (13636, 19671)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (13636, 4463) -> (13636,
13555)
     -  1 missing entry in student solution: (13636, 4463) -> (1363
6, 6279) -> (13636, 13555) -> (13636, 19671)

     -  failed on trial 1 of 500
     4
     13636  4463
     13636 19671
     13636 13555
     13636  6279

  *   5 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (11366, 6396) -> (11366, 16911)
     -  reference segment 0: (11366, 6396) -> (11366, 7442) -> (113
66, 16911) -> (11366, 17211)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (11366, 6396) -> (11366,
16911)
     -  1 missing entry in student solution: (11366, 6396) -> (1136
6, 7442) -> (11366, 16911) -> (11366, 17211)

     -  failed on trial 1 of 250
     20
     11366  6396
     19873  9430
     14914  9494
     19873  4749
     19873  5972
     19007 15789
     19873 15911
     19007  3279
```

```
    19558  3159
    11366 17211
    19558  4326
    14914  8776
    14914 15216
    19007  6090
    19007 11677
    11366  7442
    14914  1787
    19558 10327
    19558  6214
    11366 16911

  *  10 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 6: (18384, 6278) -> (18384, 18108)
     -  reference segment 7: (18384, 6278) -> (18384, 17066) -> (18
384, 18108) -> (18384, 20994)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  1 extra entry in student solution: (18384, 6278) -> (18384,
18108)
     -  1 missing entry in student solution: (18384, 6278) -> (1838
4, 17066) -> (18384, 18108) -> (18384, 20994)

     -  failed on trial 1 of 50

  *  15 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 14: (2868, 8220) -> (2868, 15178)
     -  reference segment 1: (2868, 8220) -> (2868, 14060) -> (2868
, 15178) -> (2868, 20961)

     -  student   solution has 15 non-null entries
     -  reference solution has 15 non-null entries
     -  1 extra entry in student solution: (2868, 8220) -> (2868, 1
5178)
     -  1 missing entry in student solution: (2868, 8220) -> (2868,
14060) -> (2868, 15178) -> (2868, 20961)

     -  failed on trial 1 of 5
```

```
==> FAILED

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
  *  filename = random91.txt
  *  filename = random152.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5a: Points from a file with 5 or more on some line segments
  *  filename = input9.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (1000, 1000) -> (8000, 8000)
     -  reference segment 0: (1000, 1000) -> (2000, 2000) -> (3000,
3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 6000) -> (7000, 700
0) -> (8000, 8000) -> (9000, 9000)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (1000, 1000) -> (8000, 8
000)
     -  1 missing entry in student solution: (1000, 1000) -> (2000,
2000) -> (3000, 3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 600
0) -> (7000, 7000) -> (8000, 8000) -> (9000, 9000)

  *  filename = input10.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (1000, 18000) -> (3500, 28000)
     -  reference segment 1: (1000, 18000) -> (2000, 22000) -> (300
0, 26000) -> (3500, 28000) -> (4000, 30000)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (1000, 18000) -> (3500,
28000)
     -  1 missing entry in student solution: (1000, 18000) -> (2000
, 22000) -> (3000, 26000) -> (3500, 28000) -> (4000, 30000)

  *  filename = input20.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (8192, 25088) -> (8192, 28160)
     -  reference segment 4: (8192, 25088) -> (8192, 26112) -> (819
2, 27136) -> (8192, 28160) -> (8192, 29184)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  2 extra entries in student solution, including: (4160, 2918
4) -> (7168, 29184)
     -  2 missing entries in student solution, including: (4160, 29
184) -> (5120, 29184) -> (6144, 29184) -> (7168, 29184) -> (8192, 2
9184)

  *  filename = input50.txt
  *  filename = input80.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (19000, 1000) -> (26000, 22000)
     -  reference segment 20: (19000, 1000) -> (20000, 4000) -> (26
000, 22000) -> (29000, 31000)

     -  student   solution has 31 non-null entries
     -  reference solution has 31 non-null entries
     -  3 extra entries in student solution, including: (14000, 160
00) -> (25000, 27000)
     -  3 missing entries in student solution, including: (14000, 1
6000) -> (21000, 23000) -> (25000, 27000) -> (29000, 31000)

  *  filename = input300.txt
  *  filename = inarow.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (30000, 0) -> (19000, 27500)
     -  reference segment 0: (30000, 0) -> (27000, 7500) -> (26000,
10000) -> (20000, 25000) -> (19000, 27500) -> (18000, 30000)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
```

```
    -  1 extra entry in student solution: (30000, 0) -> (19000, 27
500)
    -  1 missing entry in student solution: (30000, 0) -> (27000,
7500) -> (26000, 10000) -> (20000, 25000) -> (19000, 27500) -> (180
00, 30000)

==> FAILED

Test 5b: Points from a file with 5 or more on some line segments
  *  filename = kw1260.txt
    -  segments() contains a subsegment of a segment in reference
solution
    -  student   segment 286: (16384, 30255) -> (15169, 30414)
    -  reference segment 104: (16384, 30255) -> (15979, 30308) ->
(15574, 30361) -> (15169, 30414) -> (14764, 30467)

    -  student   solution has 288 non-null entries
    -  reference solution has 288 non-null entries
    -  2 extra entries in student solution, including: (12652, 303
95) -> (14236, 30449)
    -  2 missing entries in student solution, including: (12652, 3
0395) -> (13180, 30413) -> (13708, 30431) -> (14236, 30449) -> (147
64, 30467)

  *  filename = rs1423.txt
    -  segments() contains a subsegment of a segment in reference
solution
    -  student   segment 441: (14169, 27672) -> (13685, 27948)
    -  reference segment 127: (14169, 27672) -> (13927, 27810) ->
(13685, 27948) -> (13443, 28086)

    -  student   solution has 443 non-null entries
    -  reference solution has 443 non-null entries
    -  2 extra entries in student solution, including: (12273, 279
15) -> (13053, 28029)
    -  2 missing entries in student solution, including: (12273, 2
7915) -> (12663, 27972) -> (13053, 28029) -> (13443, 28086)

==> FAILED

Test 6: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
```

```
==> passed

Test 7: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> FAILED

Test 8: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
==> FAILED

Test 9: Random line segments, none vertical or horizontal
```

```
  *   1 random line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (801, 2000) -> (7740, 8642)
     -  reference segment 0: (801, 2000) -> (7226, 8150) -> (7740,
8642) -> (11595, 12332)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (801, 2000) -> (7740, 86
42)
     -  1 missing entry in student solution: (801, 2000) -> (7226,
8150) -> (7740, 8642) -> (11595, 12332)

     -  failed on trial 1 of 500
     4
       801  2000
     11595 12332
      7740  8642
      7226  8150

  *   5 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (412, 11869) -> (15127, 16459)
     -  reference segment 0: (412, 11869) -> (7279, 14011) -> (1512
7, 16459) -> (17089, 17071)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (412, 11869) -> (15127,
16459)
     -  1 missing entry in student solution: (412, 11869) -> (7279,
14011) -> (15127, 16459) -> (17089, 17071)

     -  failed on trial 1 of 500
     20
      9176  1403
      5057  6520
     15320 14382
     14528 12159
     13736  9936
      8490  1176
      8318   307
```

```
       7279 14011
      15127 16459
      10289  9632
       1787  4575
      19528 16187
       9522  6390
      11117  4175
       9178  4652
      11624  4008
       9823  2327
      17089 17071
       8981  8854
        412 11869

  *  25 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 18: (10082, 9912) -> (13904, 21235)
     -  reference segment 20: (10082, 9912) -> (11258, 13396) -> (1
3904, 21235) -> (14492, 22977)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (10082, 9912) -> (13904,
21235)
     -  1 missing entry in student solution: (10082, 9912) -> (1125
8, 13396) -> (13904, 21235) -> (14492, 22977)

     -  failed on trial 1 of 100

  *  50 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 26: (2941, 4584) -> (21190, 21195)
     -  reference segment 25: (2941, 4584) -> (14238, 14867) -> (21
190, 21195) -> (22059, 21986)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (2941, 4584) -> (21190,
21195)
     -  1 missing entry in student solution: (2941, 4584) -> (14238
, 14867) -> (21190, 21195) -> (22059, 21986)
```

```
     -  failed on trial 1 of 15

  *  100 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 61: (3174, 6044) -> (17469, 20414)
     -  reference segment 51: (3174, 6044) -> (10798, 13708) -> (17
469, 20414) -> (21281, 24246)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (3174, 6044) -> (17469,
20414)
     -  1 missing entry in student solution: (3174, 6044) -> (10798
, 13708) -> (17469, 20414) -> (21281, 24246)

     -  failed on trial 1 of 2

==> FAILED

Test 10: Random line segments
  *   1 random line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (10674, 4499) -> (10674, 6683)
     -  reference segment 0: (10674, 4499) -> (10674, 5279) -> (106
74, 6683) -> (10674, 6722)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (10674, 4499) -> (10674,
6683)
     -  1 missing entry in student solution: (10674, 4499) -> (1067
4, 5279) -> (10674, 6683) -> (10674, 6722)

     -  failed on trial 1 of 500
     4
     10674  4499
     10674  5279
     10674  6722
     10674  6683

  *   5 random line segments
     -  segments() contains a subsegment of a segment in reference
```

```
solution
     -  student   segment 4: (4899, 13509) -> (5307, 14563)
     -  reference segment 2: (4899, 13509) -> (5115, 14067) -> (530
7, 14563) -> (5547, 15183)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (4899, 13509) -> (5307,
14563)
     -  1 missing entry in student solution: (4899, 13509) -> (5115
, 14067) -> (5307, 14563) -> (5547, 15183)

     -  failed on trial 1 of 500
     20
      1568  7775
      6549  2960
      6639  3810
      5307 14563
      5115 14067
      5802 11951
     14310  9100
      9465  8947
      6945  6700
      6603  3470
      2882  9071
     12047  2749
     12512  4054
      4899 13509
     11025  9397
     12419  3793
     13313 10057
      5547 15183
      4488 10655
     10973  9382

  *  25 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 21: (1723, 10855) -> (8184, 13766)
     -  reference segment 7: (1723, 10855) -> (7638, 13520) -> (818
4, 13766) -> (9094, 14176)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
```

```
    -  1 extra entry in student solution: (1723, 10855) -> (8184,
13766)
    -  1 missing entry in student solution: (1723, 10855) -> (7638
, 13520) -> (8184, 13766) -> (9094, 14176)

    -  failed on trial 1 of 100

  *  50 random line segments
    -  segments() contains a subsegment of a segment in reference
solution
    -  student   segment 49: (1727, 14410) -> (4127, 14410)
    -  reference segment 2: (1727, 14410) -> (3887, 14410) -> (412
7, 14410) -> (4207, 14410)

    -  student   solution has 50 non-null entries
    -  reference solution has 50 non-null entries
    -  1 extra entry in student solution: (1727, 14410) -> (4127,
14410)
    -  1 missing entry in student solution: (1727, 14410) -> (3887
, 14410) -> (4127, 14410) -> (4207, 14410)

    -  failed on trial 1 of 15

  *  100 random line segments
    -  segments() contains a subsegment of a segment in reference
solution
    -  student   segment 87: (7455, 11025) -> (8543, 12657)
    -  reference segment 59: (7455, 11025) -> (7743, 11457) -> (85
43, 12657) -> (9951, 14769)

    -  student   solution has 100 non-null entries
    -  reference solution has 100 non-null entries
    -  1 extra entry in student solution: (7455, 11025) -> (8543,
12657)
    -  1 missing entry in student solution: (7455, 11025) -> (7743
, 11457) -> (8543, 12657) -> (9951, 14769)

    -  failed on trial 1 of 2

==> FAILED

Test 11: Random distinct points in a given range
  *  5 random points in a 10-by-10 grid
    -  segments() contains a subsegment of a segment in reference
```

```
solution
     -  student   segment 0: (8, 1) -> (8, 7)
     -  reference segment 0: (8, 1) -> (8, 4) -> (8, 7) -> (8, 9)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (8, 1) -> (8, 7)
     -  1 missing entry in student solution: (8, 1) -> (8, 4) -> (8
, 7) -> (8, 9)

     -  failed on trial 201 of 500
     5
         8     4
         8     9
         8     7
         1     8
         8     1

  *  10 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (7, 3) -> (7, 7)
     -  reference segment 1: (7, 3) -> (7, 6) -> (7, 7) -> (7, 9)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (7, 3) -> (7, 7)
     -  1 missing entry in student solution: (7, 3) -> (7, 6) -> (7
, 7) -> (7, 9)

     -  failed on trial 17 of 500
     10
         9     5
         0     6
         7     7
         7     6
         2     6
         8     6
         7     3
         2     0
         5     5
         7     9

  *  50 random points in a 10-by-10 grid
```

```
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 16: (1, 1) -> (8, 8)
     -  reference segment 22: (1, 1) -> (2, 2) -> (3, 3) -> (4, 4)
-> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 36 non-null entries
     -  reference solution has 36 non-null entries
     -  3 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  3 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (6, 9) -> (7, 9) -> (8, 9) -> (9, 9)

     -  failed on trial 1 of 100

  *  90 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (0, 0) -> (8, 8)
     -  reference segment 94: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)
-> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 131 non-null entries
     -  reference solution has 131 non-null entries
     -  8 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  8 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (4, 9) -> (5, 9) -> (7, 9) -> (8, 9)
-> (9, 9)

     -  failed on trial 1 of 50

  *  200 random points in a 50-by-50 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 71: (43, 4) -> (43, 43)
     -  reference segment 216: (43, 4) -> (43, 6) -> (43, 20) -> (4
3, 26) -> (43, 37) -> (43, 43) -> (43, 49)

     -  student   solution has 222 non-null entries
     -  reference solution has 222 non-null entries
     -  3 extra entries in student solution, including: (11, 49) ->
(37, 49)
     -  3 missing entries in student solution, including: (11, 49)
```

```
-> (22, 49) -> (37, 49) -> (43, 49)

     -  failed on trial 1 of 10

==> FAILED

Test 12: M*N points on an M-by-N grid
  *  3-by-3 grid
  *  4-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (2, 2)
     -  reference segment 5: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
2, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3)

  *  5-by-5 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (3, 3)
     -  reference segment 9: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3) -
> (4, 4)

     -  student   solution has 16 non-null entries
     -  reference solution has 16 non-null entries
     -  3 extra entries in student solution, including: (0, 4) -> (
3, 4)
     -  3 missing entries in student solution, including: (0, 4) ->
(1, 4) -> (2, 4) -> (3, 4) -> (4, 4)

  *  10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (0, 0) -> (8, 8)
     -  reference segment 110: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)
-> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 154 non-null entries
     -  reference solution has 154 non-null entries
     -  9 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  9 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (4, 9) -> (5, 9) -> (6, 9) -> (7, 9)
-> (8, 9) -> (9, 9)

  *  20-by-20 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 12: (0, 0) -> (18, 18)
     -  reference segment 1824: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3
) -> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9) -> (1
0, 10) -> (11, 11) -> (12, 12) -> (13, 13) -> (14, 14) -> (15, 15)
-> (16, 16) -> (17, 17) -> (18, 18) -> (19, 19)

     -  student   solution has 2446 non-null entries
     -  reference solution has 2446 non-null entries
     -  25 extra entries in student solution, including: (0, 19) ->
(18, 19)
     -  25 missing entries in student solution, including: (0, 19)
-> (1, 19) -> (2, 19) -> (3, 19) -> (4, 19) -> (5, 19) -> (6, 19) -
> (7, 19) -> (8, 19) -> (9, 19) -> (10, 19) -> (11, 19) -> (12, 19)
-> (13, 19) -> (14, 19) -> (15, 19) -> (16, 19) -> (17, 19) -> (18,
19) -> (19, 19)

  *  5-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (1, 0) -> (3, 2)
     -  reference segment 6: (1, 0) -> (2, 1) -> (3, 2) -> (4, 3)

     -  student   solution has 13 non-null entries
     -  reference solution has 13 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
3, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3)

  *  6-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (2, 0) -> (4, 2)
     -  reference segment 7: (2, 0) -> (3, 1) -> (4, 2) -> (5, 3)
     -  student   solution has 16 non-null entries
     -  reference solution has 16 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
4, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3)

  *  10-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (6, 2)
     -  reference segment 16: (0, 0) -> (3, 1) -> (6, 2) -> (9, 3)

     -  student   solution has 38 non-null entries
     -  reference solution has 38 non-null entries
     -  5 extra entries in student solution, including: (0, 3) -> (
8, 3)
     -  5 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3)
-> (8, 3) -> (9, 3)

  *  15-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 11: (2, 0) -> (10, 2)
     -  reference segment 34: (2, 0) -> (6, 1) -> (10, 2) -> (14, 3
)

     -  student   solution has 79 non-null entries
     -  reference solution has 79 non-null entries
     -  6 extra entries in student solution, including: (0, 3) -> (
13, 3)
     -  6 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3)
-> (8, 3) -> (9, 3) -> (10, 3) -> (11, 3) -> (12, 3) -> (13, 3) ->
(14, 3)

  *  25-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (16, 2)
     -  reference segment 96: (0, 0) -> (8, 1) -> (16, 2) -> (24, 3
)
```

```
     -  student   solution has 213 non-null entries
     -  reference solution has 213 non-null entries
     -  10 extra entries in student solution, including: (0, 3) ->
(23, 3)
     -  10 missing entries in student solution, including: (0, 3) -
> (1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3
) -> (8, 3) -> (9, 3) -> (10, 3) -> (11, 3) -> (12, 3) -> (13, 3) -
> (14, 3) -> (15, 3) -> (16, 3) -> (17, 3) -> (18, 3) -> (19, 3) ->
(20, 3) -> (21, 3) -> (22, 3) -> (23, 3) -> (24, 3)

==> FAILED

Test 13: Check that data type is immutable by testing whether each
method
         returns the same value, regardless of any intervening oper
ations
  *  input8.txt
  *  equidistant.txt
==> passed

Test 14: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
  *  equidistant.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
==> FAILED

Test 15: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 16: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 17: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 7/21 tests passed!

================================================================

*************************************************************************
***********
*  memory
*************************************************************************
***********

Computing memory of Point
*-----------------------------------------------------------
Running 1 total tests.

The maximum amount of memory per Point object is 32 bytes.

Student memory = 24 bytes (passed)

Total: 1/1 tests passed!

================================================================



*************************************************************************
```

```
***********
*  timing
********************************************************************
***********

Timing BruteCollinearPoints
*-----------------------------------------------------------
Running 10 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    16   0.02         680           0            680
45
=> passed    32   0.00        5456           0           5456
121
=> passed    64   0.00       43680           0          43680
301
=> passed   128   0.01      349504           0         349504
732
=> passed   256   0.09     2796160           0        2796160
1715
==> 5/5 tests passed

Test 2a-2e: Find collinear points among N/4 arbitrary line segments


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    16   0.00         780           0            780
45
=> passed    32   0.00        5833           0           5833
126
=> passed    64   0.00       45355           0          45355
308
=> passed   128   0.01      356068           0         356068
```

```
727
=> passed   256   0.04     2823026           0        2823026
1741
==> 5/5 tests passed

Total: 10/10 tests passed!

================================================================




Timing FastCollinearPoints
*------------------------------------------------------------
Running 31 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.03       12095       18490          49075
295
=> passed   128   0.02       48767       88113         224993
746
=> passed   256   0.06      195839      409258        1014355
1736
=> passed   512   0.22      784895     1878236        4541367
3969
=> passed  1024   0.64     3142649     8463118       20068885
8955
=> passed  2048   1.83    12576605    37895720       88368045
19986
==> 6/6 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (88368045 / 20068885) = 2.14
=> passed

==> 7/7 tests passed

Test 2a-2e: Find collinear points among the N points on an N-by-1 g
rid
```

```
                                                      slopeTo()
            N    time     slopeTo()   compare()  + 2*compare()
compareTo()
---------------------------------------------------------------------
----------------------------
=> passed    64   0.00        4159        3968           12095
370
=> passed   128   0.00       16511       16128           48767
866
=> passed   256   0.00       65791       65024          195839
1982
=> passed   512   0.02      262655      261120          784895
4486
=> passed  1024   0.06     1049599     1046528         3142655
10021
=> passed  2048   0.25     4196351     4190208        12576767
22013
=> passed  4096   0.37    16781311    16769024        50319359
48169
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (50319359 / 12576767) = 2.00
=> passed

==> 8/8 tests passed

Test 3a-3e: Find collinear points among the 4N points on an N/4-by-
4 grid


                                                      slopeTo()
            N    time     slopeTo()   compare()  + 2*compare()
compareTo()
---------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8423       16799           42021
714
=> passed   128   0.01       33575       63259          160093
2239
=> passed   256   0.02      134055      164539          463133
7457
=> passed   512   0.05      535719      564184         1664087
26321
=> passed  1024   0.10     2141863     2112673         6367209
```

```
97374
=> passed  2048   0.32     8565415     8190306       24946027
371544
=> passed  4096   1.25    34257575    32242499       98742573
1446243
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (98742573 / 24946027) = 1.98
=> passed

==> 8/8 tests passed

Test 4a-4e: Find collinear points among the 8N points on an N/8-by-
8 grid

                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8471       17829          44129
694
=> passed   128   0.00       33823       81682         197187
2157
=> passed   256   0.01      135087      324185         783457
7109
=> passed   512   0.04      539919      884726        2309371
24967
=> passed  1024   0.13     2158695     3283170        8725035
91908
=> passed  2048   0.46     8632807    12693139       34019085
349746
=> passed  4096   1.85    34527191    49915570      134358331
1358891
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (134358331 / 34019085) = 1.9
8
=> passed

==> 8/8 tests passed

Total: 31/31 tests passed!
```

```
=================================================================
```

Submission

Submission time	Thu-24-Sep 13:23:16
Raw Score	76.22 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  26/41 tests passed
Memory:       1/1 tests passed
Timing:       41/41 tests passed

Aggregate score: 76.22% [Correctness: 65%, Memory: 10%, Timing: 25%
, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 24K
-rw-r--r-- 1 2.9K Sep 24 20:23 BruteCollinearPoints.java
-rw-r--r-- 1 4.1K Sep 24 20:23 FastCollinearPoints.java
-rw-r--r-- 1 4.6K Sep 24 20:23 Point.java
-rw-r--r-- 1 4.0K Sep 24 20:23 studentSubmission.zip


*******************************************************************
***********
*  compiling
```

```
******************************************************************
***********


% javac Point.java
*-----------------------------------------------------------


================================================================

% javac BruteCollinearPoints.java
*-----------------------------------------------------------


================================================================

% javac FastCollinearPoints.java
*-----------------------------------------------------------


================================================================



% checkstyle *.java
*-----------------------------------------------------------
================================================================


% findbugs *.class
*-----------------------------------------------------------
================================================================


Testing the APIs of your programs.
*-----------------------------------------------------------
Point:

BruteCollinearPoints:

FastCollinearPoints:

================================================================


******************************************************************
***********
```

```
*  correctness
********************************************************************
***********

Testing methods in Point
*-----------------------------------------------------------
Running 3 total tests.

Test 1: p.slopeTo(q)
  *  positive infinite slope, where p and q have coordinates in [0,
500)
  *  positive infinite slope, where p and q have coordinates in [0,
32768)
  *  negative infinite slope, where p and q have coordinates in [0,
500)
  *  negative infinite slope, where p and q have coordinates in [0,
32768)
  *  positive zero     slope, where p and q have coordinates in [0,
500)
  *  positive zero     slope, where p and q have coordinates in [0,
32768)
  *  symmetric  for random points p and q with coordinates in [0, 5
00)
  *  symmetric  for random points p and q with coordinates in [0, 3
2768)
  *  transitive for random points p, q, and r with coordinates in [
0, 500)
  *  transitive for random points p, q, and r with coordinates in [
0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 500)
  *  slopeTo(), where p and q have coordinates in [0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 10)
  *  throw a java.lang.NullPointerException if argument is null
==> passed

Test 2: p.compareTo(q)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p and q have coordinates in [0, 500)
  *  antisymmetric, where p and q have coordinates in [0, 32768)
  *  transitive, where p, q, and r have coordinates in [0, 500)
  *  transitive, where p, q, and r have coordinates in [0, 32768)
  *  sign of compareTo(), where p and q have coordinates in [0, 500
)
```

```
  *  sign of compareTo(), where p and q have coordinates in [0, 327
68)
  *  sign of compareTo(), where p and q have coordinates in [0, 10)
  *  throw java.lang.NullPointerException exception if argument is
null
==> passed

Test 3: p.slopeOrder().compare(q, r)
  *  reflexive, where p and q have coordinates in [0, 500)
     Failed on trial 1 of 100000
     p                             = (303, 124)
     q                             = (13, 64)
     p.slopeOrder().compare(q, q))  = 1
  *  reflexive, where p and q have coordinates in [0, 32768)
     Failed on trial 1 of 100000
     p                             = (23048, 31111)
     q                             = (7120, 9771)
     p.slopeOrder().compare(q, q))  = 1
  *  antisymmetric, where p, q, and r have coordinates in [0, 500)
     Failed on trial 1198 of 100000
     p                            = (176, 319)
     q                            = (38, 181)
     r                            = (290, 433)
     p.slopeOrder().compare(q, r)  = 1
     p.slopeOrder().compare(r, q)  = 1
  *  antisymmetric, where p, q, and r have coordinates in [0, 32768
)
  *  transitive, where p, q, r, and s have coordinates in [0, 500)
  *  transitive, where p, q, r, and s have coordinates in [0, 32768
)
  *  sign of compare(), where p, q, and r have coordinates in [0, 5
00)
     Failed on trial 12019 of 100000
     p                        = (139, 76)
     q                        = (162, 76)
     r                        = (422, 76)
     student   p.compare(q, r) = 1
     reference p.compare(q, r) = 0
     reference p.slopeTo(q)    = 0.0
     reference p.slopeTo(r)    = 0.0
  *  sign of compare(), where p, q, and r have coordinates in [0, 3
2768)
  *  sign of compare(), where p, q, and r have coordinates in [0, 1
0)
```

```
     Failed on trial 71 of 100000
     p                          = (9, 1)
     q                          = (4, 7)
     r                          = (4, 7)
     student   p.compare(q, r) = 1
     reference p.compare(q, r) = 0
     reference p.slopeTo(q)    = -1.2
     reference p.slopeTo(r)    = -1.2
  *  throw java.lang.NullPointerException if either argument is nul
l
==> FAILED


Total: 2/3 tests passed!

================================================================

******************************************************************
***********
*  correctness (using reference Point.java and LineSegment.java)
******************************************************************
***********

Testing methods in BruteCollinearPoints
*-----------------------------------------------------------
Running 17 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - no 5 (or more) points are collinear
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
==> passed
```

```
Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 6: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed
```

```
Test 7: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 8: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 9: Random line segments
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 10: Check that data type is immutable by testing whether each
method
        returns the same value, regardless of any intervening opera
tions
  *  input8.txt
  *  equidistant.txt
==> passed

Test 11: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
  *  equidistant.txt
==> passed

Test 12: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed
```

```
Test 13: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 14: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 17/17 tests passed!

================================================================

Testing methods in FastCollinearPoints
*-----------------------------------------------------------
Running 21 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (3000, 4000) -> (14000, 15000)
     -  reference segment 1: (3000, 4000) -> (6000, 7000) -> (14000
, 15000) -> (20000, 21000)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (3000, 4000) -> (14000,
15000)
     -  1 missing entry in student solution: (3000, 4000) -> (6000,
7000) -> (14000, 15000) -> (20000, 21000)

  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> FAILED

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (2682, 14118) -> (7453, 14118)
     -  reference segment 4: (2682, 14118) -> (5067, 14118) -> (745
3, 14118) -> (7821, 14118)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (2682, 14118) -> (7453,
14118)
     -  1 missing entry in student solution: (2682, 14118) -> (5067
, 14118) -> (7453, 14118) -> (7821, 14118)

  *  filename = horizontal25.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 24: (8784, 20913) -> (16352, 20913)
     -  reference segment 24: (8784, 20913) -> (9880, 20913) -> (16
352, 20913) -> (19666, 20913)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (8784, 20913) -> (16352,
20913)
     -  1 missing entry in student solution: (8784, 20913) -> (9880
, 20913) -> (16352, 20913) -> (19666, 20913)

  *  filename = horizontal50.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 49: (5249, 20754) -> (14800, 20754)
     -  reference segment 49: (5249, 20754) -> (5559, 20754) -> (14
800, 20754) -> (17428, 20754)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (5249, 20754) -> (14800,
20754)
     -  1 missing entry in student solution: (5249, 20754) -> (5559
, 20754) -> (14800, 20754) -> (17428, 20754)

  *  filename = horizontal75.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 74: (1536, 20976) -> (14178, 20976)
     -  reference segment 74: (1536, 20976) -> (6545, 20976) -> (14
178, 20976) -> (14591, 20976)

     -  student   solution has 75 non-null entries
     -  reference solution has 75 non-null entries
     -  1 extra entry in student solution: (1536, 20976) -> (14178,
20976)
     -  1 missing entry in student solution: (1536, 20976) -> (6545
, 20976) -> (14178, 20976) -> (14591, 20976)

  *  filename = horizontal100.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 99: (5835, 20698) -> (16154, 20698)
     -  reference segment 99: (5835, 20698) -> (7673, 20698) -> (16
154, 20698) -> (19642, 20698)
```

```
     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (5835, 20698) -> (16154,
20698)
     -  1 missing entry in student solution: (5835, 20698) -> (7673
, 20698) -> (16154, 20698) -> (19642, 20698)

==> FAILED

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (9494, 19881) -> (15266, 19881)
     -  reference segment 0: (9494, 19881) -> (10054, 19881) -> (15
266, 19881) -> (15448, 19881)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (9494, 19881) -> (15266,
19881)
     -  1 missing entry in student solution: (9494, 19881) -> (1005
4, 19881) -> (15266, 19881) -> (15448, 19881)

     -  failed on trial 1 of 500
     4
      9494 19881
     15448 19881
     10054 19881
     15266 19881

  *   5 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (1780, 16249) -> (11272, 16249)
     -  reference segment 4: (1780, 16249) -> (5431, 16249) -> (112
72, 16249) -> (15699, 16249)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (1780, 16249) -> (11272,
16249)
     -  1 missing entry in student solution: (1780, 16249) -> (5431
, 16249) -> (11272, 16249) -> (15699, 16249)
```

```
     -  failed on trial 1 of 250
     20
     14930  8990
      3824 13823
      2613  8990
     13058  8990
     15699 16249
      5400  7121
      9244 14733
     17538  7121
      3717 13823
     10322 13823
      1780 16249
     11272 16249
      5709 13823
      6583 14733
     10885  7121
      5431 16249
     10956 14733
     19865 14733
      4129  7121
     12431  8990

  *  10 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference 
solution
     -  student   segment 9: (8367, 19659) -> (19649, 19659)
     -  reference segment 9: (8367, 19659) -> (14218, 19659) -> (19
649, 19659) -> (20865, 19659)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  1 extra entry in student solution: (8367, 19659) -> (19649,
19659)
     -  1 missing entry in student solution: (8367, 19659) -> (1421
8, 19659) -> (19649, 19659) -> (20865, 19659)

     -  failed on trial 1 of 50

  *  15 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference 
solution
     -  student   segment 14: (1708, 20986) -> (7771, 20986)
     -  reference segment 14: (1708, 20986) -> (3297, 20986) -> (77
71, 20986) -> (19727, 20986)

     -  student   solution has 15 non-null entries
     -  reference solution has 15 non-null entries
     -  1 extra entry in student solution: (1708, 20986) -> (7771,
20986)
     -  1 missing entry in student solution: (1708, 20986) -> (3297
, 20986) -> (7771, 20986) -> (19727, 20986)

     -  failed on trial 1 of 5

==> FAILED

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 2: (5757, 3426) -> (5757, 16647)
     -  reference segment 1: (5757, 3426) -> (5757, 13581) -> (5757
, 16647) -> (5757, 20856)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (5757, 3426) -> (5757, 1
6647)
     -  1 missing entry in student solution: (5757, 3426) -> (5757,
13581) -> (5757, 16647) -> (5757, 20856)

  *  filename = vertical25.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 22: (13536, 9107) -> (13536, 13165)
     -  reference segment 14: (13536, 9107) -> (13536, 9393) -> (13
536, 13165) -> (13536, 20946)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (13536, 9107) -> (13536,
13165)
     -  1 missing entry in student solution: (13536, 9107) -> (1353
6, 9393) -> (13536, 13165) -> (13536, 20946)

  *  filename = vertical50.txt
```

```
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (10695, 1287) -> (10695, 20756)
     -  reference segment 27: (10695, 1287) -> (10695, 10521) -> (1
0695, 20756) -> (10695, 20927)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (10695, 1287) -> (10695,
20756)
     -  1 missing entry in student solution: (10695, 1287) -> (1069
5, 10521) -> (10695, 20756) -> (10695, 20927)

  *  filename = vertical75.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 45: (18293, 5438) -> (18293, 19756)
     -  reference segment 66: (18293, 5438) -> (18293, 17680) -> (1
8293, 19756) -> (18293, 20983)

     -  student   solution has 75 non-null entries
     -  reference solution has 75 non-null entries
     -  1 extra entry in student solution: (18293, 5438) -> (18293,
19756)
     -  1 missing entry in student solution: (18293, 5438) -> (1829
3, 17680) -> (18293, 19756) -> (18293, 20983)

  *  filename = vertical100.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 84: (19597, 8445) -> (19597, 17520)
     -  reference segment 93: (19597, 8445) -> (19597, 10925) -> (1
9597, 17520) -> (19597, 20918)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (19597, 8445) -> (19597,
17520)
     -  1 missing entry in student solution: (19597, 8445) -> (1959
7, 10925) -> (19597, 17520) -> (19597, 20918)

==> FAILED

Test 3b: Random vertical line segments
```

```
  *   1 random vertical line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (8121, 6125) -> (8121, 9051)
     -  reference segment 0: (8121, 6125) -> (8121, 7549) -> (8121,
9051) -> (8121, 14002)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (8121, 6125) -> (8121, 9
051)
     -  1 missing entry in student solution: (8121, 6125) -> (8121,
7549) -> (8121, 9051) -> (8121, 14002)

     -  failed on trial 1 of 500
     4
      8121  6125
      8121 14002
      8121  9051
      8121  7549

  *   5 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (5962, 1944) -> (5962, 19097)
     -  reference segment 1: (5962, 1944) -> (5962, 9759) -> (5962,
19097) -> (5962, 19399)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (5962, 1944) -> (5962, 1
9097)
     -  1 missing entry in student solution: (5962, 1944) -> (5962,
9759) -> (5962, 19097) -> (5962, 19399)

     -  failed on trial 1 of 250
     20
      5962  9759
      5723 14384
     19176  5638
      5962 19399
     10378 19228
     12559  6871
      5723  8737
```

```
    12559 12362
    12559  1730
     5962 19097
    19176 13099
    10378 14091
    10378 18041
    12559  6208
    10378  4360
     5962  1944
     5723  9446
    19176  4998
     5723  4811
    19176 10323

  *  10 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 7: (7557, 8346) -> (7557, 17093)
     -  reference segment 2: (7557, 8346) -> (7557, 13893) -> (7557
, 17093) -> (7557, 20699)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  1 extra entry in student solution: (7557, 8346) -> (7557, 1
7093)
     -  1 missing entry in student solution: (7557, 8346) -> (7557,
13893) -> (7557, 17093) -> (7557, 20699)

     -  failed on trial 1 of 50

  *  15 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 13: (7880, 12651) -> (7880, 19202)
     -  reference segment 3: (7880, 12651) -> (7880, 15331) -> (788
0, 19202) -> (7880, 19374)

     -  student   solution has 15 non-null entries
     -  reference solution has 15 non-null entries
     -  1 extra entry in student solution: (7880, 12651) -> (7880,
19202)
     -  1 missing entry in student solution: (7880, 12651) -> (7880
, 15331) -> (7880, 19202) -> (7880, 19374)
```

```
     -  failed on trial 1 of 5

==> FAILED

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
  *  filename = random91.txt
  *  filename = random152.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5a: Points from a file with 5 or more on some line segments
  *  filename = input9.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (1000, 1000) -> (8000, 8000)
     -  reference segment 0: (1000, 1000) -> (2000, 2000) -> (3000,
3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 6000) -> (7000, 700
0) -> (8000, 8000) -> (9000, 9000)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (1000, 1000) -> (8000, 8
000)
     -  1 missing entry in student solution: (1000, 1000) -> (2000,
2000) -> (3000, 3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 600
0) -> (7000, 7000) -> (8000, 8000) -> (9000, 9000)

  *  filename = input10.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (1000, 18000) -> (3500, 28000)
     -  reference segment 1: (1000, 18000) -> (2000, 22000) -> (300
0, 26000) -> (3500, 28000) -> (4000, 30000)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (1000, 18000) -> (3500, 
28000)
     -  1 missing entry in student solution: (1000, 18000) -> (2000
, 22000) -> (3000, 26000) -> (3500, 28000) -> (4000, 30000)

  *  filename = input20.txt
     -  segments() contains a subsegment of a segment in reference 
solution
     -  student   segment 3: (8192, 25088) -> (8192, 28160)
     -  reference segment 4: (8192, 25088) -> (8192, 26112) -> (819
2, 27136) -> (8192, 28160) -> (8192, 29184)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  2 extra entries in student solution, including: (4160, 2918
4) -> (7168, 29184)
     -  2 missing entries in student solution, including: (4160, 29
184) -> (5120, 29184) -> (6144, 29184) -> (7168, 29184) -> (8192, 2
9184)

  *  filename = input50.txt
  *  filename = input80.txt
     -  segments() contains a subsegment of a segment in reference 
solution
     -  student   segment 5: (19000, 1000) -> (26000, 22000)
     -  reference segment 20: (19000, 1000) -> (20000, 4000) -> (26
000, 22000) -> (29000, 31000)

     -  student   solution has 31 non-null entries
     -  reference solution has 31 non-null entries
     -  3 extra entries in student solution, including: (14000, 160
00) -> (25000, 27000)
     -  3 missing entries in student solution, including: (14000, 1
6000) -> (21000, 23000) -> (25000, 27000) -> (29000, 31000)

  *  filename = input300.txt
  *  filename = inarow.txt
     -  segments() contains a subsegment of a segment in reference 
solution
     -  student   segment 4: (30000, 0) -> (19000, 27500)
     -  reference segment 0: (30000, 0) -> (27000, 7500) -> (26000,
10000) -> (20000, 25000) -> (19000, 27500) -> (18000, 30000)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (19000, 27
500)
     -  1 missing entry in student solution: (30000, 0) -> (27000,
7500) -> (26000, 10000) -> (20000, 25000) -> (19000, 27500) -> (180
00, 30000)

==> FAILED

Test 5b: Points from a file with 5 or more on some line segments
  *  filename = kw1260.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 286: (16384, 30255) -> (15169, 30414)
     -  reference segment 104: (16384, 30255) -> (15979, 30308) ->
(15574, 30361) -> (15169, 30414) -> (14764, 30467)

     -  student   solution has 288 non-null entries
     -  reference solution has 288 non-null entries
     -  2 extra entries in student solution, including: (12652, 303
95) -> (14236, 30449)
     -  2 missing entries in student solution, including: (12652, 3
0395) -> (13180, 30413) -> (13708, 30431) -> (14236, 30449) -> (147
64, 30467)

  *  filename = rs1423.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 441: (14169, 27672) -> (13685, 27948)
     -  reference segment 127: (14169, 27672) -> (13927, 27810) ->
(13685, 27948) -> (13443, 28086)

     -  student   solution has 443 non-null entries
     -  reference solution has 443 non-null entries
     -  2 extra entries in student solution, including: (12273, 279
15) -> (13053, 28029)
     -  2 missing entries in student solution, including: (12273, 2
7915) -> (12663, 27972) -> (13053, 28029) -> (13443, 28086)

==> FAILED

Test 6: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 7: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> FAILED

Test 8: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
==> FAILED
```

```
Test 9: Random line segments, none vertical or horizontal
  *   1 random line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (9288, 5676) -> (10998, 9871)
     -  reference segment 0: (9288, 5676) -> (10656, 9032) -> (1099
8, 9871) -> (11682, 11549)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (9288, 5676) -> (10998,
9871)
     -  1 missing entry in student solution: (9288, 5676) -> (10656
, 9032) -> (10998, 9871) -> (11682, 11549)

     -  failed on trial 1 of 500
     4
      9288  5676
     10656  9032
     11682 11549
     10998  9871

  *   5 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (3999, 8278) -> (6750, 12492)
     -  reference segment 2: (3999, 8278) -> (5964, 11288) -> (6750
, 12492) -> (11859, 20318)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (3999, 8278) -> (6750, 1
2492)
     -  1 missing entry in student solution: (3999, 8278) -> (5964,
11288) -> (6750, 12492) -> (11859, 20318)

     -  failed on trial 1 of 500
     20
     10378 11372
      5964 11288
      3999  8278
      6750 12492
      5689 11859
     10004  9488
```

```
    10687    75
     1708  6964
     7217 13545
    14533 14494
     1105  6801
    11592  4355
    17953 16502
    11687 17966
    11859 20318
    13040 11203
    11411  3499
     4925 11016
     5983  9474
     9256  5720

  *  25 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 8: (1302, 2597) -> (15642, 18947)
     -  reference segment 15: (1302, 2597) -> (5126, 6957) -> (1564
2, 18947) -> (16120, 19492)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (1302, 2597) -> (15642,
18947)
     -  1 missing entry in student solution: (1302, 2597) -> (5126,
6957) -> (15642, 18947) -> (16120, 19492)

     -  failed on trial 1 of 100

  *  50 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 32: (1874, 7926) -> (10519, 19964)
     -  reference segment 33: (1874, 7926) -> (2539, 8852) -> (1051
9, 19964) -> (11849, 21816)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (1874, 7926) -> (10519,
19964)
     -  1 missing entry in student solution: (1874, 7926) -> (2539,
8852) -> (10519, 19964) -> (11849, 21816)
```

```
     -  failed on trial 1 of 15

  *  100 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 14: (8230, 808) -> (16135, 16159)
     -  reference segment 69: (8230, 808) -> (11950, 8032) -> (1613
5, 16159) -> (18925, 21577)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (8230, 808) -> (16135, 1
6159)
     -  1 missing entry in student solution: (8230, 808) -> (11950,
8032) -> (16135, 16159) -> (18925, 21577)

     -  failed on trial 1 of 2

==> FAILED

Test 10: Random line segments
  *   1 random line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (4168, 6050) -> (5981, 6148)
     -  reference segment 0: (4168, 6050) -> (4945, 6092) -> (5981,
6148) -> (6795, 6192)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (4168, 6050) -> (5981, 6
148)
     -  1 missing entry in student solution: (4168, 6050) -> (4945,
6092) -> (5981, 6148) -> (6795, 6192)

     -  failed on trial 1 of 500
     4
      6795  6192
      4168  6050
      4945  6092
      5981  6148

  *   5 random line segments
```

```
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (6587, 12718) -> (7845, 13075)
     -  reference segment 0: (6587, 12718) -> (7623, 13012) -> (784
5, 13075) -> (12433, 14377)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (6587, 12718) -> (7845,
13075)
     -  1 missing entry in student solution: (6587, 12718) -> (7623
, 13012) -> (7845, 13075) -> (12433, 14377)

     -  failed on trial 1 of 500
     20
     12056  9134
      6039  3352
      7623 13012
      9896  6254
      4003 13003
      2237 10620
      4297 13101
     11096  7854
      7128  3726
      4811 12076
      5935 13647
      1742 10340
     12433 14377
      4591 13199
      4455  2808
      6587 12718
      4712 12020
      7845 13075
     10856  7534
     12573  5596

  *  25 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 24: (6648, 13683) -> (9457, 14690)
     -  reference segment 4: (6648, 13683) -> (8715, 14424) -> (945
7, 14690) -> (9510, 14709)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (6648, 13683) -> (9457,
14690)
     -  1 missing entry in student solution: (6648, 13683) -> (8715
, 14424) -> (9457, 14690) -> (9510, 14709)

     -  failed on trial 1 of 100

  *  50 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 44: (11782, 12107) -> (12322, 13655)
     -  reference segment 39: (11782, 12107) -> (12217, 13354) -> (
12322, 13655) -> (12592, 14429)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (11782, 12107) -> (12322
, 13655)
     -  1 missing entry in student solution: (11782, 12107) -> (122
17, 13354) -> (12322, 13655) -> (12592, 14429)

     -  failed on trial 1 of 15

  *  100 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 63: (4097, 8034) -> (5627, 15324)
     -  reference segment 84: (4097, 8034) -> (5338, 13947) -> (562
7, 15324) -> (5644, 15405)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (4097, 8034) -> (5627, 1
5324)
     -  1 missing entry in student solution: (4097, 8034) -> (5338,
13947) -> (5627, 15324) -> (5644, 15405)

     -  failed on trial 1 of 2

==> FAILED

Test 11: Random distinct points in a given range
  *  5 random points in a 10-by-10 grid
```

```
    -  segments() contains a subsegment of a segment in reference
solution
    -  student   segment 0: (1, 1) -> (3, 3)
    -  reference segment 0: (1, 1) -> (2, 2) -> (3, 3) -> (6, 6)

    -  student   solution has 1 non-null entries
    -  reference solution has 1 non-null entries
    -  1 extra entry in student solution: (1, 1) -> (3, 3)
    -  1 missing entry in student solution: (1, 1) -> (2, 2) -> (3
, 3) -> (6, 6)

    -  failed on trial 206 of 500
    5
        1     1
        3     5
        2     2
        3     3
        6     6

  *  10 random points in a 10-by-10 grid
    -  segments() contains a subsegment of a segment in reference
solution
    -  student   segment 0: (6, 0) -> (6, 5)
    -  reference segment 0: (6, 0) -> (6, 4) -> (6, 5) -> (6, 8)

    -  student   solution has 1 non-null entries
    -  reference solution has 1 non-null entries
    -  1 extra entry in student solution: (6, 0) -> (6, 5)
    -  1 missing entry in student solution: (6, 0) -> (6, 4) -> (6
, 5) -> (6, 8)

    -  failed on trial 42 of 500
    10
        4     0
        1     3
        2     3
        6     0
        0     4
        5     3
        6     8
        6     5
        2     7
        6     4
  *  50 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (8, 8)
     -  reference segment 25: (0, 0) -> (1, 1) -> (6, 6) -> (7, 7)
-> (8, 8) -> (9, 9)

     -  student   solution has 43 non-null entries
     -  reference solution has 43 non-null entries
     -  4 extra entries in student solution, including: (2, 9) -> (
8, 9)
     -  4 missing entries in student solution, including: (2, 9) ->
(6, 9) -> (8, 9) -> (9, 9)

     -  failed on trial 1 of 100

  *  90 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (0, 0) -> (8, 8)
     -  reference segment 94: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)
-> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 130 non-null entries
     -  reference solution has 130 non-null entries
     -  8 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  8 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (5, 9) -> (6, 9) -> (7, 9) -> (8, 9)
-> (9, 9)

     -  failed on trial 1 of 50

  *  200 random points in a 50-by-50 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 25: (33, 1) -> (33, 45)
     -  reference segment 202: (33, 1) -> (33, 9) -> (33, 39) -> (3
3, 45) -> (33, 49)

     -  student   solution has 212 non-null entries
     -  reference solution has 212 non-null entries
     -  4 extra entries in student solution, including: (45, 37) ->
(34, 48)
```

```
     -  4 missing entries in student solution, including: (45, 37)
-> (37, 45) -> (34, 48) -> (33, 49)

     -  failed on trial 1 of 10

==> FAILED

Test 12: M*N points on an M-by-N grid
  *  3-by-3 grid
  *  4-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (2, 2)
     -  reference segment 5: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
2, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3)

  *  5-by-5 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (3, 3)
     -  reference segment 9: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3) -
> (4, 4)

     -  student   solution has 16 non-null entries
     -  reference solution has 16 non-null entries
     -  3 extra entries in student solution, including: (0, 4) -> (
3, 4)
     -  3 missing entries in student solution, including: (0, 4) ->
(1, 4) -> (2, 4) -> (3, 4) -> (4, 4)

  *  10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (0, 0) -> (8, 8)
     -  reference segment 110: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)
-> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 154 non-null entries
     -  reference solution has 154 non-null entries
     -  9 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  9 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (4, 9) -> (5, 9) -> (6, 9) -> (7, 9)
-> (8, 9) -> (9, 9)

  *  20-by-20 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 12: (0, 0) -> (18, 18)
     -  reference segment 1824: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3
) -> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9) -> (1
0, 10) -> (11, 11) -> (12, 12) -> (13, 13) -> (14, 14) -> (15, 15)
-> (16, 16) -> (17, 17) -> (18, 18) -> (19, 19)

     -  student   solution has 2446 non-null entries
     -  reference solution has 2446 non-null entries
     -  25 extra entries in student solution, including: (0, 19) ->
(18, 19)
     -  25 missing entries in student solution, including: (0, 19)
-> (1, 19) -> (2, 19) -> (3, 19) -> (4, 19) -> (5, 19) -> (6, 19) -
> (7, 19) -> (8, 19) -> (9, 19) -> (10, 19) -> (11, 19) -> (12, 19)
-> (13, 19) -> (14, 19) -> (15, 19) -> (16, 19) -> (17, 19) -> (18,
19) -> (19, 19)

  *  5-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (1, 0) -> (3, 2)
     -  reference segment 6: (1, 0) -> (2, 1) -> (3, 2) -> (4, 3)

     -  student   solution has 13 non-null entries
     -  reference solution has 13 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
3, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3)

  *  6-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (2, 0) -> (4, 2)
     -  reference segment 7: (2, 0) -> (3, 1) -> (4, 2) -> (5, 3)
     -  student   solution has 16 non-null entries
     -  reference solution has 16 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
4, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3)

  *  10-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (6, 2)
     -  reference segment 16: (0, 0) -> (3, 1) -> (6, 2) -> (9, 3)

     -  student   solution has 38 non-null entries
     -  reference solution has 38 non-null entries
     -  5 extra entries in student solution, including: (0, 3) -> (
8, 3)
     -  5 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3)
-> (8, 3) -> (9, 3)

  *  15-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 11: (2, 0) -> (10, 2)
     -  reference segment 34: (2, 0) -> (6, 1) -> (10, 2) -> (14, 3
)

     -  student   solution has 79 non-null entries
     -  reference solution has 79 non-null entries
     -  6 extra entries in student solution, including: (0, 3) -> (
13, 3)
     -  6 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3)
-> (8, 3) -> (9, 3) -> (10, 3) -> (11, 3) -> (12, 3) -> (13, 3) ->
(14, 3)

  *  25-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (16, 2)
     -  reference segment 96: (0, 0) -> (8, 1) -> (16, 2) -> (24, 3
)
```

```
     -  student   solution has 213 non-null entries
     -  reference solution has 213 non-null entries
     -  10 extra entries in student solution, including: (0, 3) ->
(23, 3)
     -  10 missing entries in student solution, including: (0, 3) -
> (1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3
) -> (8, 3) -> (9, 3) -> (10, 3) -> (11, 3) -> (12, 3) -> (13, 3) -
> (14, 3) -> (15, 3) -> (16, 3) -> (17, 3) -> (18, 3) -> (19, 3) ->
(20, 3) -> (21, 3) -> (22, 3) -> (23, 3) -> (24, 3)

==> FAILED

Test 13: Check that data type is immutable by testing whether each
method
         returns the same value, regardless of any intervening oper
ations
  *  input8.txt
  *  equidistant.txt
==> passed

Test 14: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
  *  equidistant.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
==> FAILED

Test 15: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 16: Throws exception either if argument to constructor is null
```

```
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 17: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 7/21 tests passed!

================================================================

*******************************************************************
***********
*  memory
*******************************************************************
***********

Computing memory of Point
*-----------------------------------------------------------
Running 1 total tests.

The maximum amount of memory per Point object is 32 bytes.

Student memory = 24 bytes (passed)

Total: 1/1 tests passed!

================================================================
*********************************************************************
***********
*  timing
*********************************************************************
***********

Timing BruteCollinearPoints
*-------------------------------------------------------------
Running 10 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                         slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
---------------------------------------------------------------------
-----------------------------
=> passed    16   0.02         680           0            680
45
=> passed    32   0.00        5456           0           5456
122
=> passed    64   0.00       43680           0          43680
309
=> passed   128   0.01      349504           0         349504
739
=> passed   256   0.08     2796160           0        2796160
1738
==> 5/5 tests passed

Test 2a-2e: Find collinear points among N/4 arbitrary line segments


                                                         slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
---------------------------------------------------------------------
-----------------------------
=> passed    16   0.00         780           0            780
47
=> passed    32   0.00        5805           0           5805
116
=> passed    64   0.00       45204           0          45204
302
```

```
=> passed   128   0.01      355932           0         355932
733
=> passed   256   0.04     2822135           0        2822135
1732
==> 5/5 tests passed

Total: 10/10 tests passed!

================================================================




Timing FastCollinearPoints
*-----------------------------------------------------------
Running 31 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
-------------------------------------------------------------------
----------------------------
=> passed    64   0.03       12095       18225          48545
308
=> passed   128   0.02       48767       88419         225605
740
=> passed   256   0.06      195839      410331        1016501
1730
=> passed   512   0.21      784871     1877113        4539097
3985
=> passed  1024   0.62     3142649     8481589       20105827
8952
=> passed  2048   1.82    12576665    37907228       88391121
19925
==> 6/6 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (88391121 / 20105827) = 2.14
=> passed

==> 7/7 tests passed

Test 2a-2e: Find collinear points among the N points on an N-by-1 g
```

```
rid

                                                                 slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        4159        3968          12095
374
=> passed   128   0.00       16511       16128          48767
867
=> passed   256   0.00       65791       65024         195839
1979
=> passed   512   0.02      262655      261120         784895
4487
=> passed  1024   0.06     1049599     1046528        3142655
10003
=> passed  2048   0.17     4196351     4190208       12576767
22015
=> passed  4096   0.38    16781311    16769024       50319359
48200
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (50319359 / 12576767) = 2.00
=> passed

==> 8/8 tests passed

Test 3a-3e: Find collinear points among the 4N points on an N/4-by-
4 grid

                                                                 slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8423       16799          42021
715
=> passed   128   0.01       33575       63259         160093
2238
=> passed   256   0.01      134055      164539         463133
7457
=> passed   512   0.05      535719      564184        1664087
26329
```

```
=> passed  1024   0.13     2141863     2112673         6367209
97373
=> passed  2048   0.33     8565415     8190306        24946027
371611
=> passed  4096   1.32    34257575    32242499        98742573
1446321
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (98742573 / 24946027) = 1.98
=> passed

==> 8/8 tests passed

Test 4a-4e: Find collinear points among the 8N points on an N/8-by-
8 grid

                                                       slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8471       17829           44129
690
=> passed   128   0.00       33823       81682          197187
2156
=> passed   256   0.02      135087      324185          783457
7118
=> passed   512   0.05      539919      884726         2309371
24967
=> passed  1024   0.13     2158695     3283170         8725035
91939
=> passed  2048   0.46     8632807    12693139        34019085
349733
=> passed  4096   1.85    34527191    49915570       134358331
1358868
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (134358331 / 34019085) = 1.9
8
=> passed

==> 8/8 tests passed

Total: 31/31 tests passed!
```

```
================================================================
```

Submission

Submission time	Thu-24-Sep 13:13:07
Raw Score	74.63 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  25/41 tests passed
Memory:       1/1 tests passed
Timing:       41/41 tests passed

Aggregate score: 74.63% [Correctness: 65%, Memory: 10%, Timing: 25%
, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 24K
-rw-r--r-- 1 2.7K Sep 24 20:13 BruteCollinearPoints.java
-rw-r--r-- 1 4.1K Sep 24 20:13 FastCollinearPoints.java
-rw-r--r-- 1 4.6K Sep 24 20:13 Point.java
-rw-r--r-- 1 4.0K Sep 24 20:13 studentSubmission.zip


*******************************************************************
***********
```

```
*  compiling
*******************************************************************
***********


% javac Point.java
*-----------------------------------------------------------


================================================================

% javac BruteCollinearPoints.java
*-----------------------------------------------------------


================================================================

% javac FastCollinearPoints.java
*-----------------------------------------------------------


================================================================




% checkstyle *.java
*-----------------------------------------------------------
================================================================



% findbugs *.class
*-----------------------------------------------------------
================================================================



Testing the APIs of your programs.
*-----------------------------------------------------------
Point:

BruteCollinearPoints:

FastCollinearPoints:

================================================================



*******************************************************************
```

```
***********
*  correctness
*********************************************************************
***********

Testing methods in Point
*-----------------------------------------------------------
Running 3 total tests.

Test 1: p.slopeTo(q)
  *  positive infinite slope, where p and q have coordinates in [0,
500)
  *  positive infinite slope, where p and q have coordinates in [0,
32768)
  *  negative infinite slope, where p and q have coordinates in [0,
500)
  *  negative infinite slope, where p and q have coordinates in [0,
32768)
  *  positive zero     slope, where p and q have coordinates in [0,
500)
  *  positive zero     slope, where p and q have coordinates in [0,
32768)
  *  symmetric  for random points p and q with coordinates in [0, 5
00)
  *  symmetric  for random points p and q with coordinates in [0, 3
2768)
  *  transitive for random points p, q, and r with coordinates in [
0, 500)
  *  transitive for random points p, q, and r with coordinates in [
0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 500)
  *  slopeTo(), where p and q have coordinates in [0, 32768)
  *  slopeTo(), where p and q have coordinates in [0, 10)
  *  throw a java.lang.NullPointerException if argument is null
==> passed

Test 2: p.compareTo(q)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p and q have coordinates in [0, 500)
  *  antisymmetric, where p and q have coordinates in [0, 32768)
  *  transitive, where p, q, and r have coordinates in [0, 500)
  *  transitive, where p, q, and r have coordinates in [0, 32768)
  *  sign of compareTo(), where p and q have coordinates in [0, 500
)
  *  sign of compareTo(), where p and q have coordinates in [0, 327
68)
  *  sign of compareTo(), where p and q have coordinates in [0, 10)
  *  throw java.lang.NullPointerException exception if argument is
null
==> passed

Test 3: p.slopeOrder().compare(q, r)
  *  reflexive, where p and q have coordinates in [0, 500)
  *  reflexive, where p and q have coordinates in [0, 32768)
  *  antisymmetric, where p, q, and r have coordinates in [0, 500)
  *  antisymmetric, where p, q, and r have coordinates in [0, 32768
)
  *  transitive, where p, q, r, and s have coordinates in [0, 500)
  *  transitive, where p, q, r, and s have coordinates in [0, 32768
)
  *  sign of compare(), where p, q, and r have coordinates in [0, 5
00)
     -  wrong order: slope-ascending, but breaking ties by natural
order
     -  slope order depends only on the slope, not on the x- or y-c
oordinates
  *  sign of compare(), where p, q, and r have coordinates in [0, 3
2768)
  *  sign of compare(), where p, q, and r have coordinates in [0, 1
0)
     -  wrong order: slope-ascending, but breaking ties by natural
order
     -  slope order depends only on the slope, not on the x- or y-c
oordinates
  *  throw java.lang.NullPointerException if either argument is nul
l
==> FAILED


Total: 2/3 tests passed!

================================================================

******************************************************************
***********
*  correctness (using reference Point.java and LineSegment.java)
******************************************************************
```

```
***********

Testing methods in BruteCollinearPoints
*-----------------------------------------------------------
Running 17 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - no 5 (or more) points are collinear
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
  *  filename = horizontal25.txt
==> passed

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
  *   5 random horizontal line segments
  *  10 random horizontal line segments
  *  15 random horizontal line segments
==> passed

Test 3a: Points from a file with vertical line segments
  *  filename = vertical5.txt
  *  filename = vertical25.txt
==> passed

Test 3b: Random vertical line segments
  *   1 random vertical line segment
  *   5 random vertical line segments
  *  10 random vertical line segments
  *  15 random vertical line segments
==> passed

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 6: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 7: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
==> passed

Test 8: Random line segments, none vertical or horizontal
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed

Test 9: Random line segments
  *   1 random line segment
  *   5 random line segments
  *  10 random line segments
  *  15 random line segments
==> passed
```

```
Test 10: Check that data type is immutable by testing whether each
method
        returns the same value, regardless of any intervening opera
tions
  *  input8.txt
  *  equidistant.txt
==> passed

Test 11: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
  *  equidistant.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
==> FAILED

Test 12: numberOfSegments() is consistent with segments()
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 13: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 14: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 16/17 tests passed!

================================================================

Testing methods in FastCollinearPoints
*------------------------------------------------------------
Running 21 total tests.

The inputs satisfy the following conditions:
  - no duplicate points
  - all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file
  *  filename = input8.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (3000, 4000) -> (14000, 15000)
     -  reference segment 1: (3000, 4000) -> (6000, 7000) -> (14000
, 15000) -> (20000, 21000)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (3000, 4000) -> (14000,
15000)
     -  1 missing entry in student solution: (3000, 4000) -> (6000,
7000) -> (14000, 15000) -> (20000, 21000)

  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> FAILED

Test 2a: Points from a file with horizontal line segments
  *  filename = horizontal5.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (2682, 14118) -> (7453, 14118)
     -  reference segment 4: (2682, 14118) -> (5067, 14118) -> (745
3, 14118) -> (7821, 14118)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (2682, 14118) -> (7453,
14118)
     -  1 missing entry in student solution: (2682, 14118) -> (5067
, 14118) -> (7453, 14118) -> (7821, 14118)

  *  filename = horizontal25.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 24: (8784, 20913) -> (16352, 20913)
     -  reference segment 24: (8784, 20913) -> (9880, 20913) -> (16
352, 20913) -> (19666, 20913)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (8784, 20913) -> (16352,
20913)
     -  1 missing entry in student solution: (8784, 20913) -> (9880
, 20913) -> (16352, 20913) -> (19666, 20913)

  *  filename = horizontal50.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 49: (5249, 20754) -> (14800, 20754)
     -  reference segment 49: (5249, 20754) -> (5559, 20754) -> (14
800, 20754) -> (17428, 20754)
```

```
     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (5249, 20754) -> (14800,
20754)
     -  1 missing entry in student solution: (5249, 20754) -> (5559
, 20754) -> (14800, 20754) -> (17428, 20754)

  *  filename = horizontal75.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 74: (1536, 20976) -> (14178, 20976)
     -  reference segment 74: (1536, 20976) -> (6545, 20976) -> (14
178, 20976) -> (14591, 20976)

     -  student   solution has 75 non-null entries
     -  reference solution has 75 non-null entries
     -  1 extra entry in student solution: (1536, 20976) -> (14178,
20976)
     -  1 missing entry in student solution: (1536, 20976) -> (6545
, 20976) -> (14178, 20976) -> (14591, 20976)

  *  filename = horizontal100.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 99: (5835, 20698) -> (16154, 20698)
     -  reference segment 99: (5835, 20698) -> (7673, 20698) -> (16
154, 20698) -> (19642, 20698)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (5835, 20698) -> (16154,
20698)
     -  1 missing entry in student solution: (5835, 20698) -> (7673
, 20698) -> (16154, 20698) -> (19642, 20698)

==> FAILED

Test 2b: Random horizontal line segments
  *   1 random horizontal line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (5216, 1425) -> (13219, 1425)
     -  reference segment 0: (5216, 1425) -> (8127, 1425) -> (13219
, 1425) -> (18905, 1425)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (5216, 1425) -> (13219,
1425)
     -  1 missing entry in student solution: (5216, 1425) -> (8127,
1425) -> (13219, 1425) -> (18905, 1425)

     -  failed on trial 1 of 500
     4
      5216  1425
      8127  1425
     18905  1425
     13219  1425

  *   5 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (4267, 15435) -> (9220, 15435)
     -  reference segment 4: (4267, 15435) -> (6951, 15435) -> (922
0, 15435) -> (17762, 15435)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (4267, 15435) -> (9220,
15435)
     -  1 missing entry in student solution: (4267, 15435) -> (6951
, 15435) -> (9220, 15435) -> (17762, 15435)

     -  failed on trial 1 of 250
     20
      9220 15435
      6951 15435
     17315  7700
     13732 15211
     19513 15211
     14277 11673
     19382  7700
      4805  5462
     17762 15435
      1230 15211
      9041  7700
      9180 15211
      6223  5462
     12682  5462
     18016 11673
      6748  5462
     19061  7700
     19878 11673
      4267 15435
      8256 11673

  *  10 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 9: (11037, 18552) -> (17829, 18552)
     -  reference segment 9: (11037, 18552) -> (12434, 18552) -> (1
7829, 18552) -> (19128, 18552)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  1 extra entry in student solution: (11037, 18552) -> (17829
, 18552)
     -  1 missing entry in student solution: (11037, 18552) -> (124
34, 18552) -> (17829, 18552) -> (19128, 18552)

     -  failed on trial 1 of 50

  *  15 random horizontal line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 14: (8633, 19984) -> (15246, 19984)
     -  reference segment 14: (8633, 19984) -> (11210, 19984) -> (1
5246, 19984) -> (19213, 19984)

     -  student   solution has 15 non-null entries
     -  reference solution has 15 non-null entries
     -  1 extra entry in student solution: (8633, 19984) -> (15246,
19984)
     -  1 missing entry in student solution: (8633, 19984) -> (1121
0, 19984) -> (15246, 19984) -> (19213, 19984)

     -  failed on trial 1 of 5

==> FAILED

Test 3a: Points from a file with vertical line segments
```

```
  *  filename = vertical5.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 2: (5757, 3426) -> (5757, 16647)
     -  reference segment 1: (5757, 3426) -> (5757, 13581) -> (5757
, 16647) -> (5757, 20856)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (5757, 3426) -> (5757, 1
6647)
     -  1 missing entry in student solution: (5757, 3426) -> (5757,
13581) -> (5757, 16647) -> (5757, 20856)

  *  filename = vertical25.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 22: (13536, 9107) -> (13536, 13165)
     -  reference segment 14: (13536, 9107) -> (13536, 9393) -> (13
536, 13165) -> (13536, 20946)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (13536, 9107) -> (13536,
13165)
     -  1 missing entry in student solution: (13536, 9107) -> (1353
6, 9393) -> (13536, 13165) -> (13536, 20946)

  *  filename = vertical50.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (10695, 1287) -> (10695, 20756)
     -  reference segment 27: (10695, 1287) -> (10695, 10521) -> (1
0695, 20756) -> (10695, 20927)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (10695, 1287) -> (10695,
20756)
     -  1 missing entry in student solution: (10695, 1287) -> (1069
5, 10521) -> (10695, 20756) -> (10695, 20927)

  *  filename = vertical75.txt
     -  segments() contains a subsegment of a segment in reference
```

```
solution
     -  student   segment 45: (18293, 5438) -> (18293, 19756)
     -  reference segment 66: (18293, 5438) -> (18293, 17680) -> (1
8293, 19756) -> (18293, 20983)

     -  student   solution has 75 non-null entries
     -  reference solution has 75 non-null entries
     -  1 extra entry in student solution: (18293, 5438) -> (18293,
19756)
     -  1 missing entry in student solution: (18293, 5438) -> (1829
3, 17680) -> (18293, 19756) -> (18293, 20983)

  *  filename = vertical100.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 84: (19597, 8445) -> (19597, 17520)
     -  reference segment 93: (19597, 8445) -> (19597, 10925) -> (1
9597, 17520) -> (19597, 20918)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (19597, 8445) -> (19597,
17520)
     -  1 missing entry in student solution: (19597, 8445) -> (1959
7, 10925) -> (19597, 17520) -> (19597, 20918)

==> FAILED

Test 3b: Random vertical line segments
  *   1 random vertical line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (15505, 4533) -> (15505, 12321)
     -  reference segment 0: (15505, 4533) -> (15505, 6538) -> (155
05, 12321) -> (15505, 20552)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (15505, 4533) -> (15505,
12321)
     -  1 missing entry in student solution: (15505, 4533) -> (1550
5, 6538) -> (15505, 12321) -> (15505, 20552)

     -  failed on trial 1 of 500
     4
     15505 12321
     15505 20552
     15505  4533
     15505  6538

  *   5 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (5491, 1615) -> (5491, 7054)
     -  reference segment 0: (5491, 1615) -> (5491, 4911) -> (5491,
7054) -> (5491, 20618)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (5491, 1615) -> (5491, 7
054)
     -  1 missing entry in student solution: (5491, 1615) -> (5491,
4911) -> (5491, 7054) -> (5491, 20618)

     -  failed on trial 1 of 250
     20
     18575  1066
     18575 12065
     18575 10307
     18435  3919
      9480  1703
      8023 12739
      8023  3853
     18575 16579
      5491  4911
      5491 20618
      8023 17957
     18435 10278
     18435  1617
      5491  7054
     18435  7080
      9480 18109
      9480  3607
      8023  2583
      9480  2818
      5491  1615

  *  10 random vertical line segments
```

```
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 2: (11615, 1723) -> (11615, 11949)
     -  reference segment 4: (11615, 1723) -> (11615, 6281) -> (116
15, 11949) -> (11615, 20156)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  1 extra entry in student solution: (11615, 1723) -> (11615,
11949)
     -  1 missing entry in student solution: (11615, 1723) -> (1161
5, 6281) -> (11615, 11949) -> (11615, 20156)

     -  failed on trial 1 of 50

  *  15 random vertical line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 13: (13109, 9200) -> (13109, 11888)
     -  reference segment 7: (13109, 9200) -> (13109, 9710) -> (131
09, 11888) -> (13109, 20893)

     -  student   solution has 15 non-null entries
     -  reference solution has 15 non-null entries
     -  1 extra entry in student solution: (13109, 9200) -> (13109,
11888)
     -  1 missing entry in student solution: (13109, 9200) -> (1310
9, 9710) -> (13109, 11888) -> (13109, 20893)

     -  failed on trial 1 of 5

==> FAILED

Test 4a: Points from a file with no line segments
  *  filename = random23.txt
  *  filename = random38.txt
  *  filename = random91.txt
  *  filename = random152.txt
==> passed

Test 4b: Random points with no line segments
  *   5 random points
  *  10 random points
  *  20 random points
  *  50 random points
==> passed

Test 5a: Points from a file with 5 or more on some line segments
  *  filename = input9.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (1000, 1000) -> (8000, 8000)
     -  reference segment 0: (1000, 1000) -> (2000, 2000) -> (3000,
3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 6000) -> (7000, 700
0) -> (8000, 8000) -> (9000, 9000)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (1000, 1000) -> (8000, 8
000)
     -  1 missing entry in student solution: (1000, 1000) -> (2000,
2000) -> (3000, 3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 600
0) -> (7000, 7000) -> (8000, 8000) -> (9000, 9000)

  *  filename = input10.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (1000, 18000) -> (3500, 28000)
     -  reference segment 1: (1000, 18000) -> (2000, 22000) -> (300
0, 26000) -> (3500, 28000) -> (4000, 30000)

     -  student   solution has 2 non-null entries
     -  reference solution has 2 non-null entries
     -  1 extra entry in student solution: (1000, 18000) -> (3500,
28000)
     -  1 missing entry in student solution: (1000, 18000) -> (2000
, 22000) -> (3000, 26000) -> (3500, 28000) -> (4000, 30000)

  *  filename = input20.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (8192, 25088) -> (8192, 28160)
     -  reference segment 4: (8192, 25088) -> (8192, 26112) -> (819
2, 27136) -> (8192, 28160) -> (8192, 29184)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  2 extra entries in student solution, including: (4160, 2918
4) -> (7168, 29184)
     -  2 missing entries in student solution, including: (4160, 29
184) -> (5120, 29184) -> (6144, 29184) -> (7168, 29184) -> (8192, 2
9184)

  *  filename = input50.txt
  *  filename = input80.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (19000, 1000) -> (26000, 22000)
     -  reference segment 20: (19000, 1000) -> (20000, 4000) -> (26
000, 22000) -> (29000, 31000)

     -  student   solution has 31 non-null entries
     -  reference solution has 31 non-null entries
     -  3 extra entries in student solution, including: (14000, 160
00) -> (25000, 27000)
     -  3 missing entries in student solution, including: (14000, 1
6000) -> (21000, 23000) -> (25000, 27000) -> (29000, 31000)

  *  filename = input300.txt
  *  filename = inarow.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (30000, 0) -> (19000, 27500)
     -  reference segment 0: (30000, 0) -> (27000, 7500) -> (26000,
10000) -> (20000, 25000) -> (19000, 27500) -> (18000, 30000)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (19000, 27
500)
     -  1 missing entry in student solution: (30000, 0) -> (27000,
7500) -> (26000, 10000) -> (20000, 25000) -> (19000, 27500) -> (180
00, 30000)

==> FAILED

Test 5b: Points from a file with 5 or more on some line segments
  *  filename = kw1260.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 286: (16384, 30255) -> (15169, 30414)
     -  reference segment 104: (16384, 30255) -> (15979, 30308) ->
```

```
(15574, 30361) -> (15169, 30414) -> (14764, 30467)

     -  student   solution has 288 non-null entries
     -  reference solution has 288 non-null entries
     -  2 extra entries in student solution, including: (12652, 303
95) -> (14236, 30449)
     -  2 missing entries in student solution, including: (12652, 3
0395) -> (13180, 30413) -> (13708, 30431) -> (14236, 30449) -> (147
64, 30467)

  *  filename = rs1423.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 441: (14169, 27672) -> (13685, 27948)
     -  reference segment 127: (14169, 27672) -> (13927, 27810) ->
(13685, 27948) -> (13443, 28086)

     -  student   solution has 443 non-null entries
     -  reference solution has 443 non-null entries
     -  2 extra entries in student solution, including: (12273, 279
15) -> (13053, 28029)
     -  2 missing entries in student solution, including: (12273, 2
7915) -> (12663, 27972) -> (13053, 28029) -> (13443, 28086)

==> FAILED

Test 6: Points from a file with fewer than 4 points
  *  filename = input1.txt
  *  filename = input2.txt
  *  filename = input3.txt
==> passed

Test 7: Check for dependence on either compareTo() or compare()
        returning { -1, +1, 0 } instead of { negative integer,
        positive integer, zero }
  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = input299.txt
==> FAILED

Test 8: Check for fragile dependence on return value of toString()
  *  filename = equidistant.txt
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (30000, 0) -> (10000, 20000)
     -  reference segment 2: (30000, 0) -> (20000, 10000) -> (10000
, 20000) -> (0, 30000)

     -  student   solution has 4 non-null entries
     -  reference solution has 4 non-null entries
     -  1 extra entry in student solution: (30000, 0) -> (10000, 20
000)
     -  1 missing entry in student solution: (30000, 0) -> (20000,
10000) -> (10000, 20000) -> (0, 30000)

  *  filename = input40.txt
  *  filename = input48.txt
==> FAILED

Test 9: Random line segments, none vertical or horizontal
  *   1 random line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (7606, 6590) -> (15735, 10748)
     -  reference segment 0: (7606, 6590) -> (14257, 9992) -> (1573
5, 10748) -> (16474, 11126)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (7606, 6590) -> (15735,
10748)
     -  1 missing entry in student solution: (7606, 6590) -> (14257
, 9992) -> (15735, 10748) -> (16474, 11126)
```

```
     -  failed on trial 1 of 500
     4
     15735 10748
     16474 11126
     14257  9992
      7606  6590

  *   5 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (1923, 597) -> (14163, 12762)
     -  reference segment 3: (1923, 597) -> (6003, 4652) -> (14163,
12762) -> (16611, 15195)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (1923, 597) -> (14163, 1
2762)
     -  1 missing entry in student solution: (1923, 597) -> (6003,
4652) -> (14163, 12762) -> (16611, 15195)

     -  failed on trial 1 of 500
     20
      1923   597
     10758 10201
      6083  4251
     10196 13416
     14163 12762
      6003  4652
      4840  2601
      7665  6351
       673  7669
     16611 15195
     14844 13871
     10587 10195
      8401  8509
     12188 13611
      8204 13221
      9781  8659
      3063  9931
      2919    51
      2329  7849
      1866  9889
  *  25 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (10099, 1051) -> (11987, 8259)
     -  reference segment 20: (10099, 1051) -> (10807, 3754) -> (11
987, 8259) -> (15055, 19972)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (10099, 1051) -> (11987,
8259)
     -  1 missing entry in student solution: (10099, 1051) -> (1080
7, 3754) -> (11987, 8259) -> (15055, 19972)

     -  failed on trial 1 of 100

  *  50 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 30: (9137, 6739) -> (15065, 19232)
     -  reference segment 33: (9137, 6739) -> (14153, 17310) -> (15
065, 19232) -> (16889, 23076)

     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (9137, 6739) -> (15065,
19232)
     -  1 missing entry in student solution: (9137, 6739) -> (14153
, 17310) -> (15065, 19232) -> (16889, 23076)

     -  failed on trial 1 of 15

  *  100 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 95: (2099, 10572) -> (6479, 16338)
     -  reference segment 53: (2099, 10572) -> (4289, 13455) -> (64
79, 16338) -> (11589, 23065)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (2099, 10572) -> (6479,
16338)
     -  1 missing entry in student solution: (2099, 10572) -> (4289
, 13455) -> (6479, 16338) -> (11589, 23065)

     -  failed on trial 1 of 2

==> FAILED

Test 10: Random line segments
  *   1 random line segment
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (1971, 11636) -> (4272, 11636)
     -  reference segment 0: (1971, 11636) -> (2829, 11636) -> (427
2, 11636) -> (5247, 11636)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (1971, 11636) -> (4272,
11636)
     -  1 missing entry in student solution: (1971, 11636) -> (2829
, 11636) -> (4272, 11636) -> (5247, 11636)

     -  failed on trial 1 of 500
     4
      1971 11636
      2829 11636
      5247 11636
      4272 11636

  *   5 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (3830, 10332) -> (5904, 12780)
     -  reference segment 2: (3830, 10332) -> (4745, 11412) -> (590
4, 12780) -> (6758, 13788)

     -  student   solution has 5 non-null entries
     -  reference solution has 5 non-null entries
     -  1 extra entry in student solution: (3830, 10332) -> (5904,
12780)
     -  1 missing entry in student solution: (3830, 10332) -> (4745
, 11412) -> (5904, 12780) -> (6758, 13788)

     -  failed on trial 1 of 500
     20
```

```
       6758 13788
       9856  8470
      12641  8577
       9732  8803
       4745 11412
      12066 10810
      11160  9245
      12641  8640
       6248  8222
       6248  6494
      11811 10540
       3830 10332
      12641  8596
      12641  8607
       6248  7502
       5904 12780
       6248  7448
      10111  8740
      11244  9271
       9354  8686

  *  25 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 15: (7840, 6662) -> (13937, 12826)
     -  reference segment 12: (7840, 6662) -> (12390, 11262) -> (13
937, 12826) -> (14938, 13838)

     -  student   solution has 25 non-null entries
     -  reference solution has 25 non-null entries
     -  1 extra entry in student solution: (7840, 6662) -> (13937,
12826)
     -  1 missing entry in student solution: (7840, 6662) -> (12390
, 11262) -> (13937, 12826) -> (14938, 13838)

     -  failed on trial 1 of 100

  *  50 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 48: (10689, 12642) -> (10689, 13668)
     -  reference segment 46: (10689, 12642) -> (10689, 12669) -> (
10689, 13668) -> (10689, 14370)
```

```
     -  student   solution has 50 non-null entries
     -  reference solution has 50 non-null entries
     -  1 extra entry in student solution: (10689, 12642) -> (10689
, 13668)
     -  1 missing entry in student solution: (10689, 12642) -> (106
89, 12669) -> (10689, 13668) -> (10689, 14370)

     -  failed on trial 1 of 15

  *  100 random line segments
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 93: (5787, 12199) -> (6723, 13798)
     -  reference segment 73: (5787, 12199) -> (6675, 13716) -> (67
23, 13798) -> (7467, 15069)

     -  student   solution has 100 non-null entries
     -  reference solution has 100 non-null entries
     -  1 extra entry in student solution: (5787, 12199) -> (6723,
13798)
     -  1 missing entry in student solution: (5787, 12199) -> (6675
, 13716) -> (6723, 13798) -> (7467, 15069)

     -  failed on trial 1 of 2

==> FAILED

Test 11: Random distinct points in a given range
  *  5 random points in a 10-by-10 grid
  *  10 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 0: (8, 3) -> (8, 5)
     -  reference segment 0: (8, 3) -> (8, 4) -> (8, 5) -> (8, 9)

     -  student   solution has 1 non-null entries
     -  reference solution has 1 non-null entries
     -  1 extra entry in student solution: (8, 3) -> (8, 5)
     -  1 missing entry in student solution: (8, 3) -> (8, 4) -> (8
, 5) -> (8, 9)

     -  failed on trial 4 of 500
     10
         0     3
         8     9
         9     0
         8     4
         4     1
         9     1
         6     2
         8     3
         3     7
         8     5

  *  50 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 6: (4, 0) -> (4, 8)
     -  reference segment 35: (4, 0) -> (4, 1) -> (4, 3) -> (4, 5)
-> (4, 6) -> (4, 7) -> (4, 8) -> (4, 9)

     -  student   solution has 39 non-null entries
     -  reference solution has 39 non-null entries
     -  3 extra entries in student solution, including: (0, 9) -> (
3, 9)
     -  3 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (3, 9) -> (4, 9)

     -  failed on trial 1 of 100

  *  90 random points in a 10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (0, 0) -> (8, 8)
     -  reference segment 84: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)
-> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 124 non-null entries
     -  reference solution has 124 non-null entries
     -  8 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  8 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (4, 9) -> (5, 9) -> (6, 9) -> (7, 9)
-> (8, 9) -> (9, 9)

     -  failed on trial 1 of 50

  *  200 random points in a 50-by-50 grid
```

```
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 8: (20, 0) -> (44, 42)
     -  reference segment 161: (20, 0) -> (36, 28) -> (44, 42) -> (
48, 49)

     -  student   solution has 215 non-null entries
     -  reference solution has 215 non-null entries
     -  4 extra entries in student solution, including: (6, 49) ->
(44, 49)
     -  4 missing entries in student solution, including: (6, 49) -
> (23, 49) -> (31, 49) -> (44, 49) -> (48, 49)

     -  failed on trial 1 of 10

==> FAILED

Test 12: M*N points on an M-by-N grid
  *  3-by-3 grid
  *  4-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (2, 2)
     -  reference segment 5: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)

     -  student   solution has 10 non-null entries
     -  reference solution has 10 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
2, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3)

  *  5-by-5 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (3, 3)
     -  reference segment 9: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3) -
> (4, 4)

     -  student   solution has 16 non-null entries
     -  reference solution has 16 non-null entries
     -  3 extra entries in student solution, including: (0, 4) -> (
3, 4)
     -  3 missing entries in student solution, including: (0, 4) ->
```

```
(1, 4) -> (2, 4) -> (3, 4) -> (4, 4)

  *  10-by-10 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 4: (0, 0) -> (8, 8)
     -  reference segment 110: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3)
-> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)

     -  student   solution has 154 non-null entries
     -  reference solution has 154 non-null entries
     -  9 extra entries in student solution, including: (0, 9) -> (
8, 9)
     -  9 missing entries in student solution, including: (0, 9) ->
(1, 9) -> (2, 9) -> (3, 9) -> (4, 9) -> (5, 9) -> (6, 9) -> (7, 9)
-> (8, 9) -> (9, 9)

  *  20-by-20 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 12: (0, 0) -> (18, 18)
     -  reference segment 1824: (0, 0) -> (1, 1) -> (2, 2) -> (3, 3
) -> (4, 4) -> (5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9) -> (1
0, 10) -> (11, 11) -> (12, 12) -> (13, 13) -> (14, 14) -> (15, 15)
-> (16, 16) -> (17, 17) -> (18, 18) -> (19, 19)

     -  student   solution has 2446 non-null entries
     -  reference solution has 2446 non-null entries
     -  25 extra entries in student solution, including: (0, 19) ->
(18, 19)
     -  25 missing entries in student solution, including: (0, 19)
-> (1, 19) -> (2, 19) -> (3, 19) -> (4, 19) -> (5, 19) -> (6, 19) -
> (7, 19) -> (8, 19) -> (9, 19) -> (10, 19) -> (11, 19) -> (12, 19)
-> (13, 19) -> (14, 19) -> (15, 19) -> (16, 19) -> (17, 19) -> (18,
19) -> (19, 19)

  *  5-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 3: (1, 0) -> (3, 2)
     -  reference segment 6: (1, 0) -> (2, 1) -> (3, 2) -> (4, 3)

     -  student   solution has 13 non-null entries
     -  reference solution has 13 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
3, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3)

  *  6-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 5: (2, 0) -> (4, 2)
     -  reference segment 7: (2, 0) -> (3, 1) -> (4, 2) -> (5, 3)

     -  student   solution has 16 non-null entries
     -  reference solution has 16 non-null entries
     -  3 extra entries in student solution, including: (0, 3) -> (
4, 3)
     -  3 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3)

  *  10-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (6, 2)
     -  reference segment 16: (0, 0) -> (3, 1) -> (6, 2) -> (9, 3)

     -  student   solution has 38 non-null entries
     -  reference solution has 38 non-null entries
     -  5 extra entries in student solution, including: (0, 3) -> (
8, 3)
     -  5 missing entries in student solution, including: (0, 3) ->
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3)
-> (8, 3) -> (9, 3)

  *  15-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 11: (2, 0) -> (10, 2)
     -  reference segment 34: (2, 0) -> (6, 1) -> (10, 2) -> (14, 3
)

     -  student   solution has 79 non-null entries
     -  reference solution has 79 non-null entries
     -  6 extra entries in student solution, including: (0, 3) -> (
13, 3)
     -  6 missing entries in student solution, including: (0, 3) ->
```

```
(1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3)
-> (8, 3) -> (9, 3) -> (10, 3) -> (11, 3) -> (12, 3) -> (13, 3) ->
(14, 3)

  *  25-by-4 grid
     -  segments() contains a subsegment of a segment in reference
solution
     -  student   segment 1: (0, 0) -> (16, 2)
     -  reference segment 96: (0, 0) -> (8, 1) -> (16, 2) -> (24, 3
)

     -  student   solution has 213 non-null entries
     -  reference solution has 213 non-null entries
     -  10 extra entries in student solution, including: (0, 3) ->
(23, 3)
     -  10 missing entries in student solution, including: (0, 3) -
> (1, 3) -> (2, 3) -> (3, 3) -> (4, 3) -> (5, 3) -> (6, 3) -> (7, 3
) -> (8, 3) -> (9, 3) -> (10, 3) -> (11, 3) -> (12, 3) -> (13, 3) -
> (14, 3) -> (15, 3) -> (16, 3) -> (17, 3) -> (18, 3) -> (19, 3) ->
(20, 3) -> (21, 3) -> (22, 3) -> (23, 3) -> (24, 3)

==> FAILED

Test 13: Check that data type is immutable by testing whether each
method
         returns the same value, regardless of any intervening oper
ations
  *  input8.txt
  *  equidistant.txt
==> passed

Test 14: Check that data type does not mutate the constructor argum
ent
  *  input8.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
  *  equidistant.txt
     -  data type mutated the points[] array
     -  data type should have no side effects unless documented in
API
==> FAILED

Test 15: numberOfSegments() is consistent with segments()
```

```
  *  filename = input8.txt
  *  filename = equidistant.txt
  *  filename = input40.txt
  *  filename = input48.txt
  *  filename = horizontal5.txt
  *  filename = vertical5.txt
  *  filename = random23.txt
==> passed

Test 16: Throws exception either if argument to constructor is null
         or if any entry in array is null
  *  argument is null
  *  Point[] of length 10, number of null entries = 1
  *  Point[] of length 10, number of null entries = 10
  *  Point[] of length 4, number of null entries = 1
  *  Point[] of length 3, number of null entries = 1
  *  Point[] of length 2, number of null entries = 1
  *  Point[] of length 1, number of null entries = 1
==> passed

Test 17: Constructor throws exception if duplicate points
  *  20 points
  *  10 points
  *  5 points
  *  4 points
  *  3 points
  *  2 points
==> passed


Total: 7/21 tests passed!

================================================================

*******************************************************************
***********
*  memory
*******************************************************************
***********

Computing memory of Point
*-----------------------------------------------------------
Running 1 total tests.
```

```
The maximum amount of memory per Point object is 32 bytes.

Student memory = 24 bytes (passed)

Total: 1/1 tests passed!

================================================================



********************************************************************
***********
*  timing
********************************************************************
***********

Timing BruteCollinearPoints
*------------------------------------------------------------
Running 10 total tests.

Test 1a-1e: Find collinear points among N random distinct points


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    16   0.02         680           0            680
43
=> passed    32   0.00        5456           0           5456
119
=> passed    64   0.00       43680           0          43680
307
=> passed   128   0.01      349504           0         349504
733
=> passed   256   0.04     2796160           0        2796160
1729
==> 5/5 tests passed

Test 2a-2e: Find collinear points among N/4 arbitrary line segments


                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    16   0.00         759           0            759
43
=> passed    32   0.00        5790           0           5790
122
=> passed    64   0.00       45213           0          45213
300
=> passed   128   0.01      355630           0         355630
736
=> passed   256   0.04     2823948           0        2823948
1719
==> 5/5 tests passed

Total: 10/10 tests passed!

=================================================================




Timing FastCollinearPoints
*-------------------------------------------------------------
Running 31 total tests.

Test 1a-1e: Find collinear points among N random distinct points



                                                         slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.03       12095       18243          48581
303
=> passed   128   0.02       48767       87298         223363
733
=> passed   256   0.03      195839      410149        1016137
1731
=> passed   512   0.25      784895     1879656        4544207
3980
=> passed  1024   0.64     3142637     8476681       20095999
8979
```

```
=> passed  2048   1.76    12576563    37838318       88253199
19992
==> 6/6 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (88253199 / 20095999) = 2.13
=> passed

==> 7/7 tests passed

Test 2a-2e: Find collinear points among the N points on an N-by-1 g
rid

                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        4159        3968          12095
364
=> passed   128   0.00       16511       16128          48767
856
=> passed   256   0.00       65791       65024         195839
1967
=> passed   512   0.02      262655      261120         784895
4489
=> passed  1024   0.05     1049599     1046528        3142655
9980
=> passed  2048   0.25     4196351     4190208       12576767
22053
=> passed  4096   0.37    16781311    16769024       50319359
48098
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (50319359 / 12576767) = 2.00
=> passed

==> 8/8 tests passed

Test 3a-3e: Find collinear points among the 4N points on an N/4-by-
4 grid

                                                      slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
```

```
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8423       16799           42021
714
=> passed   128   0.01       33575       63259          160093
2232
=> passed   256   0.01      134055      164539          463133
7454
=> passed   512   0.04      535719      564184         1664087
26318
=> passed  1024   0.16     2141863     2112673         6367209
97398
=> passed  2048   0.33     8565415     8190306        24946027
371536
=> passed  4096   1.33    34257575    32242499        98742573
1446258
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (98742573 / 24946027) = 1.98
=> passed

==> 8/8 tests passed

Test 4a-4e: Find collinear points among the 8N points on an N/8-by-
8 grid

                                                        slopeTo()
             N    time     slopeTo()   compare()  + 2*compare()
compareTo()
--------------------------------------------------------------------
----------------------------
=> passed    64   0.00        8471       17829           44129
702
=> passed   128   0.00       33823       81682          197187
2156
=> passed   256   0.01      135087      324185          783457
7139
=> passed   512   0.04      539919      884726         2309371
24965
=> passed  1024   0.13     2158695     3283170         8725035
91914
=> passed  2048   0.47     8632807    12693139        34019085
349717
=> passed  4096   1.88    34527191    49915570       134358331
```

```
1358886
==> 7/7 tests passed

lg ratio(slopeTo() + 2*compare()) = lg (134358331 / 34019085) = 1.9
8
=> passed

==> 8/8 tests passed

Total: 31/31 tests passed!

================================================================
```