See the Assessment Guide for information on how to interpret this report.

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     PASSED
PMD:          PASSED
Checkstyle:   FAILED (0 errors, 2 warnings)

Correctness:  33/35 tests passed
Memory:       16/16 tests passed
Timing:       42/42 tests passed

Aggregate score: 96.57%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 60%, Timing: 10%, Memory: 20% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
9.5K Apr  8 18:26 KdTree.java
2.4K Apr  8 18:26 PointSET.java


================================================================================
*  COMPILING
================================================================================


% javac PointSET.java
*-----------------------------------------------------------

% javac KdTree.java
*-----------------------------------------------------------


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


Checking the APIs of your programs.
*-----------------------------------------------------------
PointSET:

KdTree:

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


================================================================================
*  CHECKING STYLE AND COMMON BUG PATTERNS
================================================================================


% spotbugs *.class
*-----------------------------------------------------------


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


% pmd .
*-----------------------------------------------------------


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


% checkstyle *.java
*-----------------------------------------------------------
[WARN] KdTree.java:56:21: Using a quadruple nested if statement suggests poor design in this program. [NestedIfDepth]
```

```
[WARN] KdTree.java:232:25: Using a quadruple nested if statement suggests poor design in this program. [NestedIfDepth]
Checkstyle ends with 0 errors and 2 warnings.

% custom checkstyle checks for PointSET.java
*-----------------------------------------------------------

% custom checkstyle checks for KdTree.java
*-----------------------------------------------------------


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


********************************************************************************
*  TESTING CORRECTNESS
********************************************************************************

Testing correctness of PointSET
*-----------------------------------------------------------
Running 8 total tests.

A point in an m-by-m grid means that it is of the form (i/m, j/m),
where i and j are integers between 0 and m

Test 1: insert n random points; check size() and isEmpty() after each insertion
        (size may be less than n because of duplicates)
  * 5 random points in a 1-by-1 grid
  * 50 random points in a 8-by-8 grid
  * 100 random points in a 16-by-16 grid
  * 1000 random points in a 128-by-128 grid
  * 5000 random points in a 1024-by-1024 grid
  * 50000 random points in a 65536-by-65536 grid
==> passed

Test 2: insert n random points; check contains() with random query points
  * 1 random points in a 1-by-1 grid
  * 10 random points in a 4-by-4 grid
  * 20 random points in a 8-by-8 grid
  * 10000 random points in a 128-by-128 grid
  * 100000 random points in a 1024-by-1024 grid
  * 100000 random points in a 65536-by-65536 grid
==> passed

Test 3: insert random points; check nearest() with random query points
  * 10 random points in a 4-by-4 grid
  * 15 random points in a 8-by-8 grid
  * 20 random points in a 16-by-16 grid
  * 100 random points in a 32-by-32 grid
  * 10000 random points in a 65536-by-65536 grid
==> passed

Test 4: insert random points; check range() with random query rectangles
  * 2 random points and random rectangles in a 2-by-2 grid
  * 10 random points and random rectangles in a 4-by-4 grid
  * 20 random points and random rectangles in a 8-by-8 grid
  * 100 random points and random rectangles in a 16-by-16 grid
  * 1000 random points and random rectangles in a 64-by-64 grid
  * 10000 random points and random rectangles in a 128-by-128 grid
==> passed

Test 5: call methods before inserting any points
 * size() and isEmpty()
 * contains()
 * nearest()
 * range()
==> passed

Test 6: call methods with null argument
  * insert()
  * contains()
  * range()
  * nearest()
==> passed

Test 7: check intermixed sequence of calls to insert(), isEmpty(),
        size(), contains(), range(), and nearest() with
        probabilities (p1, p2, p3, p4, p5, p6, p7), respectively
  * 10000 calls with random points in a 1-by-1 grid
    and probabilities (0.3, 0.1, 0.1, 0.1, 0.2, 0.2)
  * 10000 calls with random points in a 16-by-16 grid
    and probabilities (0.3, 0.1, 0.1, 0.1, 0.2, 0.2)
  * 10000 calls with random points in a 128-by-128 grid
    and probabilities (0.3, 0.1, 0.1, 0.1, 0.2, 0.2)
  * 10000 calls with random points in a 1024-by-1024 grid
```

```
    and probabilities (0.3, 0.1, 0.1, 0.1, 0.2, 0.2)
  * 10000 calls with random points in a 8192-by-8192 grid
    and probabilities (0.3, 0.1, 0.1, 0.1, 0.2, 0.2)
  * 10000 calls with random points in a 65536-by-65536 grid
    and probabilities (0.3, 0.1, 0.1, 0.1, 0.2, 0.2)
==> passed

Test 8: check that two PointSET objects can be created at the same time
==> passed


Total: 8/8 tests passed!


================================================================
Testing correctness of KdTree
*----------------------------------------------------------------
Running 27 total tests.

In the tests below, we consider three classes of points and rectangles.

  * Non-degenerate points: no two points (or rectangles) share either an
                           x-coordinate or a y-coordinate

  * Distinct points:       no two points (or rectangles) share both an
                           x-coordinate and a y-coordinate

  * General points:        no restrictions on the x-coordinates or y-coordinates
                           of the points (or rectangles)

A point in an m-by-m grid means that it is of the form (i/m, j/m),
where i and j are integers between 0 and m (inclusive).

Test 1a: insert points from file; check size() and isEmpty() after each insertion
  * input0.txt
  * input1.txt
  * input5.txt
  * input10.txt
==> passed

Test 1b: insert non-degenerate points; check size() and isEmpty() after each insertion
  * 1 random non-degenerate points in a 1-by-1 grid
  * 5 random non-degenerate points in a 8-by-8 grid
  * 10 random non-degenerate points in a 16-by-16 grid
  * 50 random non-degenerate points in a 128-by-128 grid
  * 500 random non-degenerate points in a 1024-by-1024 grid
  * 50000 random non-degenerate points in a 65536-by-65536 grid
==> passed

Test 1c: insert distinct points; check size() and isEmpty() after each insertion
  * 1 random distinct points in a 1-by-1 grid
  * 10 random distinct points in a 8-by-8 grid
  * 20 random distinct points in a 16-by-16 grid
  * 10000 random distinct points in a 128-by-128 grid
  * 100000 random distinct points in a 1024-by-1024 grid
  * 100000 random distinct points in a 65536-by-65536 grid
==> passed

Test 1d: insert general points; check size() and isEmpty() after each insertion
  * 5 random general points in a 1-by-1 grid
  * 10 random general points in a 4-by-4 grid
  * 50 random general points in a 8-by-8 grid
  * 100000 random general points in a 16-by-16 grid
  * 100000 random general points in a 128-by-128 grid
  * 100000 random general points in a 1024-by-1024 grid
==> passed

Test 2a: insert points from file; check contains() with random query points
  * input0.txt
  * input1.txt
  * input5.txt
  * input10.txt
==> passed

Test 2b: insert non-degenerate points; check contains() with random query points
  * 1 random non-degenerate points in a 1-by-1 grid
  * 5 random non-degenerate points in a 8-by-8 grid
  * 10 random non-degenerate points in a 16-by-16 grid
  * 20 random non-degenerate points in a 32-by-32 grid
  * 500 random non-degenerate points in a 1024-by-1024 grid
  * 10000 random non-degenerate points in a 65536-by-65536 grid
==> passed

Test 2c: insert distinct points; check contains() with random query points
```

```
  * 1 random distinct points in a 1-by-1 grid
  * 10 random distinct points in a 4-by-4 grid
  * 20 random distinct points in a 8-by-8 grid
  * 10000 random distinct points in a 128-by-128 grid
  * 100000 random distinct points in a 1024-by-1024 grid
  * 100000 random distinct points in a 65536-by-65536 grid
==> passed

Test 2d: insert general points; check contains() with random query points
  * 10000 random general points in a 1-by-1 grid
  * 10000 random general points in a 16-by-16 grid
  * 10000 random general points in a 128-by-128 grid
  * 10000 random general points in a 1024-by-1024 grid
==> passed

Test 3a: insert points from file; check range() with random query rectangles
  * input0.txt
  * input1.txt
  * input5.txt
  * input10.txt
==> passed

Test 3b: insert non-degenerate points; check range() with random query rectangles
  * 1 random non-degenerate points and random rectangles in a 2-by-2 grid
  * 5 random non-degenerate points and random rectangles in a 8-by-8 grid
  * 10 random non-degenerate points and random rectangles in a 16-by-16 grid
  * 20 random non-degenerate points and random rectangles in a 32-by-32 grid
  * 500 random non-degenerate points and random rectangles in a 1024-by-1024 grid
  * 10000 random non-degenerate points and random rectangles in a 65536-by-65536 grid
==> passed

Test 3c: insert distinct points; check range() with random query rectangles
  * 2 random distinct points and random rectangles in a 2-by-2 grid
  * 10 random distinct points and random rectangles in a 4-by-4 grid
  * 20 random distinct points and random rectangles in a 8-by-8 grid
  * 100 random distinct points and random rectangles in a 16-by-16 grid
  * 1000 random distinct points and random rectangles in a 64-by-64 grid
  * 10000 random distinct points and random rectangles in a 128-by-128 grid
==> passed

Test 3d: insert general points; check range() with random query rectangles
  * 5000 random general points and random rectangles in a 2-by-2 grid
  * 5000 random general points and random rectangles in a 16-by-16 grid
  * 5000 random general points and random rectangles in a 128-by-128 grid
  * 5000 random general points and random rectangles in a 1024-by-1024 grid
==> passed

Test 3e: insert random points; check range() with tiny rectangles
         enclosing each point
  * 5 tiny rectangles and 5 general points in a 2-by-2 grid
  * 10 tiny rectangles and 10 general points in a 4-by-4 grid
  * 20 tiny rectangles and 20 general points in a 8-by-8 grid
  * 5000 tiny rectangles and 5000 general points in a 128-by-128 grid
  * 5000 tiny rectangles and 5000 general points in a 1024-by-1024 grid
  * 5000 tiny rectangles and 5000 general points in a 65536-by-65536 grid
==> passed

Test 4a: insert points from file; check range() with random query rectangles
         and check traversal of kd-tree
  * input5.txt
  * input10.txt
==> passed

Test 4b: insert non-degenerate points; check range() with random query rectangles
         and check traversal of kd-tree
  * 3 random non-degenerate points and 1000 random rectangles in a 4-by-4 grid
  * 6 random non-degenerate points and 1000 random rectangles in a 8-by-8 grid
  * 10 random non-degenerate points and 1000 random rectangles in a 16-by-16 grid
  * 20 random non-degenerate points and 1000 random rectangles in a 32-by-32 grid
  * 30 random non-degenerate points and 1000 random rectangles in a 64-by-64 grid
==> passed

Test 5a: insert points from file; check nearest() with random query points
  * input0.txt
  * input1.txt
  * input5.txt
  * input10.txt
==> passed

Test 5b: insert non-degenerate points; check nearest() with random query points
  * 5 random non-degenerate points in a 8-by-8 grid
  * 10 random non-degenerate points in a 16-by-16 grid
  * 20 random non-degenerate points in a 32-by-32 grid
  * 30 random non-degenerate points in a 64-by-64 grid
```

```
  * 10000 random non-degenerate points in a 65536-by-65536 grid
==> passed

Test 5c: insert distinct points; check nearest() with random query points
  * 10 random distinct points in a 4-by-4 grid
  * 15 random distinct points in a 8-by-8 grid
  * 20 random distinct points in a 16-by-16 grid
  * 100 random distinct points in a 32-by-32 grid
  * 10000 random distinct points in a 65536-by-65536 grid
==> passed

Test 5d: insert general points; check nearest() with random query points
  * 10000 random general points in a 16-by-16 grid
  * 10000 random general points in a 128-by-128 grid
  * 10000 random general points in a 1024-by-1024 grid
==> passed

Test 6a: insert points from file; check nearest() with random query points
         and check traversal of kd-tree
  * input5.txt
    - student   nearest() = (0.4, 0.7)
    - reference nearest() = (0.4, 0.7)
    - performs incorrect traversal of kd-tree during call to nearest()
    - query point = (0.14, 1.0)
    - sequence of points inserted:
      A  0.7 0.2
      B  0.5 0.4
      C  0.2 0.3
      D  0.4 0.7
      E  0.9 0.6
    - student sequence of kd-tree nodes involved in calls to Point2D methods:
      A B E D
    - reference sequence of kd-tree nodes involved in calls to Point2D methods:
      A B D
    - failed on trial 9 of 1000

  * input10.txt
    - student   nearest() = (0.144, 0.179)
    - reference nearest() = (0.144, 0.179)
    - performs incorrect traversal of kd-tree during call to nearest()
    - query point = (0.22, 0.02)
    - sequence of points inserted:
      A  0.372 0.497
      B  0.564 0.413
      C  0.226 0.577
      D  0.144 0.179
      E  0.083 0.51
      F  0.32 0.708
      G  0.417 0.362
      H  0.862 0.825
      I  0.785 0.725
      J  0.499 0.208
    - student sequence of kd-tree nodes involved in calls to Point2D methods:
      A C B D G E
    - reference sequence of kd-tree nodes involved in calls to Point2D methods:
      A C D E B G
    - failed on trial 1 of 1000

==> FAILED

Test 6b: insert non-degenerate points; check nearest() with random query points
         and check traversal of kd-tree
  * 5 random non-degenerate points in a 8-by-8 grid
  * 10 random non-degenerate points in a 16-by-16 grid
    - student   nearest() = (0.625, 0.3125)
    - reference nearest() = (0.625, 0.3125)
    - performs incorrect traversal of kd-tree during call to nearest()
    - query point = (0.5, 0.25)
    - sequence of points inserted:
      A  0.25 0.125
      B  0.0625 0.5
      C  0.9375 0.8125
      D  0.125 0.1875
      E  0.3125 1.0
      F  0.625 0.3125
      G  0.0 0.0
      H  0.5625 0.5625
      I  0.875 0.6875
      J  0.75 0.4375
    - student sequence of kd-tree nodes involved in calls to Point2D methods:
      A C B F H I J
    - reference sequence of kd-tree nodes involved in calls to Point2D methods:
      A C F H I J
    - failed on trial 2 of 1000
```

```
  * 20 random non-degenerate points in a 32-by-32 grid
    - student   nearest() = (0.71875, 0.34375)
    - reference nearest() = (0.71875, 0.34375)
    - performs incorrect traversal of kd-tree during call to nearest()
    - query point = (0.78125, 0.21875)
    - sequence of points inserted:
      A  0.34375 0.0625
      B  0.5625 0.5
      C  0.375 0.28125
      D  0.0625 0.25
      E  0.6875 0.09375
      F  0.4375 0.375
      G  0.46875 0.78125
      H  0.15625 0.1875
      I  0.71875 0.34375
      J  0.75 1.0
      K  0.84375 0.53125
      L  0.875 0.625
      M  0.59375 0.84375
      N  0.03125 0.8125
      O  0.1875 0.75
      P  0.9375 0.65625
      Q  0.25 0.40625
      R  0.0 0.59375
      S  0.40625 0.6875
      T  0.5 0.125
    - student sequence of kd-tree nodes involved in calls to Point2D methods:
      A B C G E F I T
    - reference sequence of kd-tree nodes involved in calls to Point2D methods:
      A B C E F I T
    - failed on trial 1 of 1000

  * 30 random non-degenerate points in a 64-by-64 grid
    - student   nearest() = (0.78125, 0.15625)
    - reference nearest() = (0.78125, 0.15625)
    - performs incorrect traversal of kd-tree during call to nearest()
    - number of student   entries = 8
    - number of reference entries = 6
    - entry 2 of the two sequences are not equal
    - student   entry 2 = (0.0625, 0.484375)
    - reference entry 2 = (0.90625, 0.390625)

    - failed on trial 1 of 1000

  * 50 random non-degenerate points in a 128-by-128 grid
    - student   nearest() = (0.2734375, 0.984375)
    - reference nearest() = (0.2734375, 0.984375)
    - performs incorrect traversal of kd-tree during call to nearest()
    - number of student   entries = 6
    - number of reference entries = 5
    - entry 2 of the two sequences are not equal
    - student   entry 2 = (0.6484375, 0.65625)
    - reference entry 2 = (0.1875, 0.890625)

    - failed on trial 1 of 1000

  * 1000 random non-degenerate points in a 2048-by-2048 grid
    - student   nearest() = (0.66552734375, 0.76806640625)
    - reference nearest() = (0.66552734375, 0.76806640625)
    - performs incorrect traversal of kd-tree during call to nearest()
    - number of student   entries = 43
    - number of reference entries = 26
    - entry 3 of the two sequences are not equal
    - student   entry 3 = (0.8349609375, 0.27490234375)
    - reference entry 3 = (0.81103515625, 0.92431640625)

    - failed on trial 1 of 1000

==> FAILED

Test 7: check with no points
  * size() and isEmpty()
  * contains()
  * nearest()
  * range()
==> passed

Test 8: check that the specified exception is thrown with null arguments
  * argument to insert() is null
  * argument to contains() is null
  * argument to range() is null
  * argument to nearest() is null
==> passed
```

```
Test 9a: check intermixed sequence of calls to insert(), isEmpty(),
         size(), contains(), range(), and nearest() with probabilities
         (p1, p2, p3, p4, p5, p6), respectively
  * 20000 calls with non-degenerate points in a 1-by-1 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with non-degenerate points in a 16-by-16 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with non-degenerate points in a 128-by-128 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with non-degenerate points in a 1024-by-1024 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with non-degenerate points in a 8192-by-8192 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with non-degenerate points in a 65536-by-65536 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
==> passed

Test 9b: check intermixed sequence of calls to insert(), isEmpty(),
         size(), contains(), range(), and nearest() with probabilities
         (p1, p2, p3, p4, p5, p6), respectively
  * 20000 calls with distinct points in a 1-by-1 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with distinct points in a 16-by-16 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with distinct points in a 128-by-128 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with distinct points in a 1024-by-1024 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with distinct points in a 8192-by-8192 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with distinct points in a 65536-by-65536 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
==> passed

Test 9c: check intermixed sequence of calls to insert(), isEmpty(),
         size(), contains(), range(), and nearest() with probabilities
         (p1, p2, p3, p4, p5, p6), respectively
  * 20000 calls with general points in a 1-by-1 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with general points in a 16-by-16 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with general points in a 128-by-128 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with general points in a 1024-by-1024 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with general points in a 8192-by-8192 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
  * 20000 calls with general points in a 65536-by-65536 grid
    and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)
==> passed

Test 10: insert n random points into two different KdTree objects;
        check that repeated calls to size(), contains(), range(),
        and nearest() with the same arguments yield same results
  * 10 random general points in a 4-by-4 grid
  * 20 random general points in a 8-by-8 grid
  * 100 random general points in a 128-by-128 grid
  * 1000 random general points in a 65536-by-65536 grid
==> passed


Total: 25/27 tests passed!


================================================================
********************************************************************************
*  MEMORY
********************************************************************************

Analyzing memory of Point2D
*-----------------------------------------------------------
Memory of Point2D object = 32 bytes

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



Analyzing memory of RectHV
*-----------------------------------------------------------
Memory of RectHV object = 48 bytes

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

```
Analyzing memory of PointSET
*-----------------------------------------------------------
Running 8 total tests.

Memory usage of a PointSET with n points (including Point2D and RectHV objects).
Maximum allowed memory is 96n + 200 bytes.

                n       student (bytes)    reference (bytes)
----------------------------------------------------------------
=> passed       1           240                264
=> passed       2           336                360
=> passed       5           624                648
=> passed      10          1104               1128
=> passed      25          2544               2568
=> passed     100          9744               9768
=> passed     400         38544              38568
=> passed     800         76944              76968
==> 8/8 tests passed

Total: 8/8 tests passed!

Estimated student   memory (bytes) = 96.00 n + 144.00  (R^2 = 1.000)
Estimated reference memory (bytes) = 96.00 n + 168.00  (R^2 = 1.000)

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



Analyzing memory of KdTree
*-----------------------------------------------------------
Running 8 total tests.

Memory usage of a KdTree with n points (including Point2D and RectHV objects).
Maximum allowed memory is 312n + 192 bytes.

                n       student (bytes)    reference (bytes)
----------------------------------------------------------------
=> passed       1           168                160
=> passed       2           304                288
=> passed       5           712                672
=> passed      10          1392               1312
=> passed      25          3432               3232
=> passed     100         13632              12832
=> passed     400         54432              51232
=> passed     800        108832             102432
==> 8/8 tests passed

Total: 8/8 tests passed!

Estimated student   memory (bytes) = 136.00 n + 32.00  (R^2 = 1.000)
Estimated reference memory (bytes) = 128.00 n + 32.00  (R^2 = 1.000)

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



********************************************************************************
*  TIMING
********************************************************************************

Timing PointSET
*-----------------------------------------------------------
Running 14 total tests.


Inserting n points into a PointSET

              n      ops per second
----------------------------------------
=> passed   160000    1366430
=> passed   320000    1370505
=> passed   640000    1154594
=> passed  1280000     823431
==> 4/4 tests passed

Performing contains() queries after inserting n points into a PointSET

              n      ops per second
----------------------------------------
=> passed   160000     639546
=> passed   320000     600153
=> passed   640000     555753
```

```
=> passed  1280000     479125
==> 4/4 tests passed

Performing range() queries after inserting n points into a PointSET

               n      ops per second
----------------------------------------
=> passed    10000       4875
=> passed    20000       1774
=> passed    40000        804
==> 3/3 tests passed

Performing nearest() queries after inserting n points into a PointSET

               n      ops per second
----------------------------------------
=> passed    10000       7399
=> passed    20000       2140
=> passed    40000        891
==> 3/3 tests passed

Total: 14/14 tests passed!


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



Timing KdTree
*------------------------------------------------------------
Running 28 total tests.


Test 1a-d: Insert n points into a 2d tree. The table gives the average number of calls
           to methods in RectHV and Point per call to insert().

                                                                                                  Point2D
               n      ops per second       RectHV()           x()                y()             equals()
-------------------------------------------------------------------------------------------------------------
=> passed   160000     838930               1.0               45.8               43.8             21.6
=> passed   320000     763913               1.0               46.6               44.6             22.0
=> passed   640000     680338               1.0               49.6               47.6             23.5
=> passed  1280000     547285               1.0               53.8               51.8             25.6
==> 4/4 tests passed


Test 2a-h: Perform contains() queries after inserting n points into a 2d tree. The table gives
           the average number of calls to methods in RectHV and Point per call to contains().

                                                                                  Point2D
               n      ops per second       x()                y()               equals()
-----------------------------------------------------------------------------------------
=> passed    10000     899041              18.5               17.5              18.0
=> passed    20000     868105              19.7               18.7              19.2
=> passed    40000     749696              21.8               20.8              21.3
=> passed    80000     632565              22.0               21.0              21.5
=> passed   160000     545501              23.2               22.2              22.7
=> passed   320000     459964              25.0               24.0              24.5
=> passed   640000     458824              25.7               24.7              25.2
=> passed  1280000     301277              27.2               26.2              26.7
==> 8/8 tests passed


Test 3a-h: Perform range() queries after inserting n points into a 2d tree. The table gives
           the average number of calls to methods in RectHV and Point per call to range().

               n      ops per second       intersects()       contains()         x()                y()
-------------------------------------------------------------------------------------------------------------
=> passed    10000     431149              50.4               31.1               50.1               12.1
=> passed    20000     419194              52.7               32.6               53.3               16.2
=> passed    40000     373802              64.9               39.3               63.1               14.1
=> passed    80000     306776              67.1               40.7               65.2               14.9
=> passed   160000     250242              70.0               42.5               70.9               20.4
=> passed   320000     217231              67.0               40.2               65.2               15.7
=> passed   640000     198107              72.0               43.3               70.7               19.2
=> passed  1280000     166478              78.7               47.0               74.8               14.2
==> 8/8 tests passed


Test 4a-h: Perform nearest() queries after inserting n points into a 2d tree. The table gives
           the average number of calls to methods in RectHV and Point per call to nearest().

                                    Point2D                RectHV
               n      ops per second     distanceSquaredTo()     distanceSquaredTo()        x()                y()
```

```
-----------------------------------------------------------------------------------------------------------------------
=> passed     10000    273155              160.2                74.2               202.2              198.5
=> passed     20000    242651              187.2                87.3               237.5              234.4
=> passed     40000    198463              238.4               111.5               309.2              300.1
=> passed     80000    179280              246.6               115.6               311.3              317.9
=> passed    160000    148297              279.3               131.4               364.9              359.3
=> passed    320000    121027              302.3               142.7               389.4              389.3
=> passed    640000     94072              314.8               148.5               404.2              408.6
=> passed   1280000     77778              379.2               179.6               496.2              486.7
==> 8/8 tests passed


Total: 28/28 tests passed!


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