ClosestPair.java

```
1    package DivideAndConquerTesting;
2
3    public class ClosestPair {
4        int numberPoints;
5        public Location[] array;
6        Location point1 = null;
7        Location point2 = null;
8        private static double minNum = Double.MAX_VALUE;
9
10       public static void setMinNum(double minNum) {
11           ClosestPair.minNum = minNum;
12       }
13
14       public static void setSecondCount(int secondCount) {
15           ClosestPair.secondCount = secondCount;
16       }
17
18       private static int secondCount = 0;
19
20       ClosestPair(int points) {
21           numberPoints = points;
22           array = new Location[numberPoints];
23       }
24
25       public static class Location {
26           double x;
27           double y;
28           Location(final double xpar, final double ypar) { // Save x, y coordinates
29               this.x = xpar;
30               this.y = ypar;
31           }
32       }
33
34       public Location[] createLocation(int numberValues) {
35 1         return new Location[numberValues];
36       }
37
38       public Location buildLocation(double x, double y) {
39 1         return new Location(x, y);
40       }
41
42       public int xPartition(final Location[] a, final int first, final int last) {
43           Location pivot = a[last]; // pivot
44 1         int i = first - 1;
45           Location temp; // Temporarily store value for position transformation
46 4         for (int j = first; j <= last - 1; j++) {
47 2             if (a[j].x <= pivot.x) { // Less than or less than pivot
48 1                 i++;
49                  temp = a[i]; // array[i] <-> array[j]
50                  a[i] = a[j];
```

```
51                      a[j] = temp;
52                  }
53              }
54  1           i++;
55              temp = a[i]; // array[pivot] <-> array[i]
56              a[i] = a[last];
57              a[last] = temp;
58  1           return i; // pivot index
59          }
60
61          public int yPartition(final Location[] a, final int first, final int last) {
62              Location pivot = a[last]; // pivot
63  1           int i = first - 1;
64              Location temp; // Temporarily store value for position transformation
65  4           for (int j = first; j <= last - 1; j++) {
66  2               if (a[j].y <= pivot.y) { // Less than or less than pivot
67  1                   i++;
68                      temp = a[i]; // array[i] <-> array[j]
69                      a[i] = a[j];
70                      a[j] = temp;
71                  }
72              }
73  1           i++;
74              temp = a[i]; // array[pivot] <-> array[i]
75              a[i] = a[last];
76              a[last] = temp;
77  1           return i; // pivot index
78          }
79
80          public void xQuickSort(
81              final Location[] a,
82              final int first,
83              final int last
84          ) {
85  2           if (first < last) {
86                  int q = xPartition(a, first, last); // pivot
87  2               xQuickSort(a, first, q - 1); // Left
88  2               xQuickSort(a, q + 1, last); // Right
89              }
90          }
91
92          public void yQuickSort(
93              final Location[] a,
94              final int first,
95              final int last
96          ) {
97  2           if (first < last) {
98                  int q = yPartition(a, first, last); // pivot
99  2               yQuickSort(a, first, q - 1); // Left
100 2               yQuickSort(a, q + 1, last); // Right
101             }
102         }
103
104         public double closestPair(final Location[] a, final int indexNum) {
```

```
105          Location[] divideArray = new Location[indexNum];
106 1        System.arraycopy(a, 0, divideArray, 0, indexNum); // Copy previous array
107 1        int divideX = indexNum / 2; // Intermediate value for divide
108          Location[] leftArray = new Location[divideX]; // divide - left array
109 1        Location[] rightArray = new Location[indexNum - divideX];
110 2        if (indexNum <= 3) { // If the number of coordinates is 3 or less
111 1            return bruteForce(divideArray);
112          }
113 1        System.arraycopy(divideArray, 0, leftArray, 0, divideX);
114 1        System.arraycopy(
115              divideArray,
116              divideX,
117              rightArray,
118              0,
119 1            indexNum - divideX
120          );
121
122          double minLeftArea; // Minimum length of left array
123          double minRightArea; // Minimum length of right array
124          double minValue; // Minimum lengt
125
126          minLeftArea = closestPair(leftArray, divideX); // recursive closestPair
127 1        minRightArea = closestPair(rightArray, indexNum - divideX);
128          minValue = Math.min(minLeftArea, minRightArea);
129 3        for (int i = 0; i < indexNum; i++) {
130 1            double xGap = Math.abs(divideArray[divideX].x - divideArray[i].x);
131 2            if (xGap < minValue) {
132 2                ClosestPair.setSecondCount(secondCount + 1); // size of the array
133              } else {
134 2                if (divideArray[i].x > divideArray[divideX].x) {
135                      break;
136                  }
137              }
138          }
139          Location[] firstWindow = new Location[secondCount];
140          int k = 0;
141 3        for (int i = 0; i < indexNum; i++) {
142 1            double xGap = Math.abs(divideArray[divideX].x - divideArray[i].x);
143 2            if (xGap < minValue) { // if it's inside a window
144                  firstWindow[k] = divideArray[i]; // put in an array
145 1                k++;
146              } else {
147 2                if (divideArray[i].x > divideArray[divideX].x) {
148                      break;
149                  }
150              }
151          }
152 2        yQuickSort(firstWindow, 0, secondCount - 1); // Sort by y coordinates
153
154          double length;
155 4        for (int i = 0; i < secondCount - 1; i++) {
156 4            for (int j = (i + 1); j < secondCount; j++) {
157 1                double xGap = Math.abs(firstWindow[i].x - firstWindow[j].x);
158 1                double yGap = Math.abs(firstWindow[i].y - firstWindow[j].y);
```

```
159 2                if (yGap < minValue) {
160 1                    length = Math.sqrt(Math.pow(xGap, 2) + Math.pow(yGap, 2));
161 2                    if (length < minValue) {
162                          minValue = length;
163 2                        if (length < minNum) {
164 1                            ClosestPair.setMinNum(length);
165                              point1 = firstWindow[i];
166                              point2 = firstWindow[j];
167                          }
168                      }
169                  } else {
170                      break;
171                  }
172              }
173          }
174 1        ClosestPair.setSecondCount(0);
175 1        return minValue;
176      }
177
178      public double bruteForce(final Location[] arrayParam) {
179          double minValue = Double.MAX_VALUE; // minimum distance
180          double length;
181          double xGap; // Difference between x coordinates
182          double yGap; // Difference between y coordinates
183          double result = 0;
184
185 1        if (arrayParam.length == 2) {
186 1            xGap = (arrayParam[0].x - arrayParam[1].x);
187 1            yGap = (arrayParam[0].y - arrayParam[1].y);
188 1            length = Math.sqrt(Math.pow(xGap, 2) + Math.pow(yGap, 2));
189 2            if (length < minNum) {
190 1                ClosestPair.setMinNum(length);
191              }
192              point1 = arrayParam[0];
193              point2 = arrayParam[1];
194              result = length;
195          }
196 1        if (arrayParam.length == 3) {
197 4            for (int i = 0; i < arrayParam.length - 1; i++) {
198 3                for (int j = (i + 1); j < arrayParam.length; j++) {
199 1                    xGap = (arrayParam[i].x - arrayParam[j].x);
200 1                    yGap = (arrayParam[i].y - arrayParam[j].y);
201 1                    length = Math.sqrt(Math.pow(xGap, 2) + Math.pow(yGap, 2));
202 2                    if (length < minValue) {
203                          minValue = length;
204 2                        if (length < minNum) {
205 1                            ClosestPair.setMinNum(length);
206                              point1 = arrayParam[i];
207                              point2 = arrayParam[j];
208                          }
209                      }
210                  }
211              }
212              result = minValue;
```

```
213             }
214 1           return result;
215         }
216     }
```

Mutations

35 1. replaced return value with null for DivideAndConquerTesting/ClosestPair::createLocation → NO_COVERAGE

39 1. replaced return value with null for DivideAndConquerTesting/ClosestPair::buildLocation → KILLED

44 1. Replaced integer subtraction with addition → KILLED

46 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED
3. Replaced integer subtraction with addition → KILLED
4. negated conditional → KILLED

47 1. changed conditional boundary → KILLED
2. negated conditional → KILLED

48 1. Changed increment from 1 to -1 → KILLED

54 1. Changed increment from 1 to -1 → KILLED

58 1. replaced int return with 0 for DivideAndConquerTesting/ClosestPair::xPartition → KILLED

63 1. Replaced integer subtraction with addition → KILLED

65 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED
3. Replaced integer subtraction with addition → KILLED
4. negated conditional → KILLED

66 1. changed conditional boundary → KILLED
2. negated conditional → KILLED

67 1. Changed increment from 1 to -1 → KILLED

73 1. Changed increment from 1 to -1 → KILLED

77 1. replaced int return with 0 for DivideAndConquerTesting/ClosestPair::yPartition → KILLED

85 1. changed conditional boundary → KILLED
2. negated conditional → KILLED

87 1. Replaced integer subtraction with addition → KILLED
2. removed call to DivideAndConquerTesting/ClosestPair::xQuickSort → KILLED

88 1. Replaced integer addition with subtraction → KILLED
2. removed call to DivideAndConquerTesting/ClosestPair::xQuickSort → KILLED

97 1. changed conditional boundary → KILLED
2. negated conditional → KILLED

99 1. Replaced integer subtraction with addition → KILLED
2. removed call to DivideAndConquerTesting/ClosestPair::yQuickSort → KILLED

100 1. Replaced integer addition with subtraction → KILLED
2. removed call to DivideAndConquerTesting/ClosestPair::yQuickSort → KILLED

106 1. removed call to java/lang/System::arraycopy → KILLED

107 1. Replaced integer division with multiplication → KILLED

109 1. Replaced integer subtraction with addition → KILLED

110 1. changed conditional boundary → KILLED
2. negated conditional → SURVIVED

111 1. replaced double return with 0.0d for DivideAndConquerTesting/ClosestPair::closestPair → KILLED

113 1. removed call to java/lang/System::arraycopy → KILLED

114 1. removed call to java/lang/System::arraycopy → KILLED

119 1. Replaced integer subtraction with addition → KILLED

127 1. Replaced integer subtraction with addition → KILLED

129 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED

	3. negated conditional → KILLED
130	1. Replaced double subtraction with addition → KILLED
131	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
132	1. Replaced integer addition with subtraction → KILLED 2. removed call to DivideAndConquerTesting/ClosestPair::setSecondCount → KILLED
134	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
141	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
142	1. Replaced double subtraction with addition → KILLED
143	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
145	1. Changed increment from 1 to -1 → KILLED
147	1. changed conditional boundary → SURVIVED 2. negated conditional → KILLED
152	1. Replaced integer subtraction with addition → KILLED 2. removed call to DivideAndConquerTesting/ClosestPair::yQuickSort → KILLED
155	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. Replaced integer subtraction with addition → KILLED 4. negated conditional → KILLED
156	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. Replaced integer addition with subtraction → KILLED 4. negated conditional → KILLED
157	1. Replaced double subtraction with addition → KILLED
158	1. Replaced double subtraction with addition → SURVIVED
159	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
160	1. Replaced double addition with subtraction → KILLED
161	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
163	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
164	1. removed call to DivideAndConquerTesting/ClosestPair::setMinNum → SURVIVED
174	1. removed call to DivideAndConquerTesting/ClosestPair::setSecondCount → KILLED
175	1. replaced double return with 0.0d for DivideAndConquerTesting/ClosestPair::closestPair → KILLED
185	1. negated conditional → KILLED
186	1. Replaced double subtraction with addition → NO_COVERAGE
187	1. Replaced double subtraction with addition → NO_COVERAGE
188	1. Replaced double addition with subtraction → NO_COVERAGE
189	1. changed conditional boundary → NO_COVERAGE 2. negated conditional → NO_COVERAGE
190	1. removed call to DivideAndConquerTesting/ClosestPair::setMinNum → NO_COVERAGE
196	1. negated conditional → KILLED
197	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. Replaced integer subtraction with addition → KILLED 4. negated conditional → KILLED
198	1. changed conditional boundary → KILLED 2. Replaced integer addition with subtraction → KILLED 3. negated conditional → SURVIVED
199	1. Replaced double subtraction with addition → KILLED
200	1. Replaced double subtraction with addition → KILLED
201	1. Replaced double addition with subtraction → KILLED
202	1. changed conditional boundary → KILLED

```
    2. negated conditional → KILLED
204 1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
205 1. removed call to DivideAndConquerTesting/ClosestPair::setMinNum → SURVIVED
214 1. replaced double return with 0.0d for
    DivideAndConquerTesting/ClosestPair::bruteForce → KILLED
```

Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

Tests examined

- DivideAndConquerTesting.AllDivideConquerTesting.[engine:junit-jupiter]/[class:DivideAndConquerTesting.AllDivideConquerTesting]/[method:testClosestPair()] (13 ms)
- DivideAndConquerTesting.AllDivideConquerTesting.[engine:junit-jupiter]/[class:DivideAndConquerTesting.AllDivideConquerTesting]/[method:BinarySearch2dArrayTestMiddle()] (28 ms)

Report generated by PIT 1.6.8