[ all classes ] [ <empty package name> ]

# Coverage Summary for Class: Bisect (<empty package name>)

| Class | Method, % | Line, % |
|---|---|---|
| Bisect | 100% (9/ 9) | 100% (38/ 38) |
| Bisect$polynomial | | |
| Bisect$RootNotFound | 100% (1/ 1) | 100% (1/ 1) |
| **total** | 100% (10/ 10) | 100% (39/ 39) |

```
 1  public class Bisect {
 2
 3     private double tolerance;
 4     private int maxIterations;
 5     private polynomial func;
 6
 7     public Bisect(polynomial f) {
 8           func = f;
 9        tolerance = 0.000001;
10        maxIterations = 50;
11     }
12
13     public Bisect(double tol, polynomial f) {
14           func = f;
15        tolerance = tol;
16        maxIterations = 50;
17     }
18
19     public Bisect(int maxIter, polynomial f) {
20           func = f;
21        tolerance = 0.000001;
22        maxIterations = maxIter;
23     }
24
25     public Bisect(double tol, int maxIter, polynomial f) {
26           func = f;
27        tolerance = tol;
28        maxIterations = maxIter;
29     }
30
31     public double getTolerance() {
32        return tolerance;
33     }
34
35     public void setTolerance(double tol) {
36        if (tol > 0)
37           tolerance = tol;
38     }
39
40     public double getMaxIterations() {
41        return maxIterations;
42     }
43
44     public void setMaxIterations(int maxIter) {
45        if (maxIter > 0)
46           maxIterations = maxIter;
47     }
48
49
50     public double run(double x1, double x2) throws RootNotFound {
51
52        int iterNum = 1;
53        double f1, f2, fmid;
54        double mid = 0;
55
56        do {
57           f1 = func.eval(x1);
58           f2 = func.eval(x2);
59
60           if (f1 * f2 > 0) {
61              throw new RootNotFound();
```

```
62          }
63
64          mid = (x1 + x2) / 2;
65          fmid = func.eval(mid);
66          if (fmid * f1 < 0)
67             x2 = mid;
68          else
69             x1 = mid;
70          iterNum++;
71       } while (Math.abs(x1 - x2) / 2 >= tolerance && Math.abs(fmid) > tolerance && iterNum <= maxIterations);
72
73       if (iterNum >= maxIterations) {
74           throw new RootNotFound();
75       }
76
77       return mid;
78    }
79
80    public interface polynomial {
81           public double eval(double value);
82    }
83
84    public class RootNotFound extends Exception {
85
86    }
87 }
```

generated on 2019-11-05 17:29