

**Ein weiteres Einreichungsbeispiel zum
"JUnit-Backend"
- LittleMath.java -**

Christian Baumann

16. Januar 2008

Inhaltsverzeichnis

1 Fehlerlose Beispieleinreichung:	3
2 Einreichung mit syntaktischem Fehler:	5
3 Einreichung mit semantischem Fehler:	7
4 Unit-Tests:	9

1 Fehlerlose Beispielinreichung:

```
1 public class LittleMath {
2     /**
3      * Returns absolute value of type long.
4      * @param l Long input
5      * @return Absolute value of l.
6      */
7     public long abs(long l){
8         if(l < 0) return -l;
9         return l;
10    }
11
12    /**
13     * Returns absolute value of type double.
14     * @param d Double input
15     * @return Absolute Value of d.
16     */
17    public double abs(double d){
18        if(d < 0) return -d;
19        return d;
20    }
21
22    /**
23     * Returns absolute value of type float.
24     * @param f Float input
25     * @return Absolute Value of f.
26     */
27    public float abs(float f){
28        return (float)abs((double)f);
29    }
30
31    /**
32     * Returns absolute value of type int.
33     * @param i Int input
34     * @return Absolute Value of i.
35     */
36    public int abs(int i){
37        return (int)abs((long)i);
38    }
39
40    /**
41     * Returns the maximum of two given long values.
42     * @param l1 Long input.
43     * @param l2 Long input.
44     * @return Maximum of l1 and l2.
45     */
46    public long max(long l1, long l2){
47        if (l1 < l2) return l2;
48        return l1;
49    }
50
51    /**
52     * Returns the maximum of two given int values.
53     * @param i1 Int input.
54     * @param i2 Int input.
55     * @return Maximum of i1 and i2.
56     */
57    public int max(int i1, int i2){
58        return (int)max((long)i1, (long)i2);
59    }
60
61    /**
62     * Returns the maximum of two given double values.
63     * @param d1 Double input.
```

```

64     * @param d2 Double input.
65     * @return Maximum of d1 and d2.
66     */
67     public double max(double d1, double d2){
68         if (d1 < d2) return d2;
69         return d1;
70     }
71
72     /**
73     * Returns the maximum of two given float values.
74     * @param f1 Float input.
75     * @param f2 Float input.
76     * @return Maximum of f1 and f2.
77     */
78     public double max(float f1, float f2){
79         return (float)max((double)f1, (double)f2);
80     }
81
82     /**
83     * Returns the minimum of two given long values.
84     * @param l1 Long input.
85     * @param l2 Long input.
86     * @return Minimum of l1 and l2.
87     */
88     public long min(long l1, long l2){
89         if (l1 < l2) return l1;
90         return l2;
91     }
92
93     /**
94     * Returns the minimum of two given int values.
95     * @param i1 Int input.
96     * @param i2 Int input.
97     * @return Minimum of i1 and i2.
98     */
99     public int min(int i1, int i2){
100        return (int)min((long)i1, (long)i2);
101    }
102
103    /**
104    * Returns the minimum of two given double values.
105    * @param d1 Double input.
106    * @param d2 Double input.
107    * @return Minimum of d1 and d2.
108    */
109    public double min(double d1, double d2){
110        if (d1 < d2) return d1;
111        return d2;
112    }
113
114    /**
115    * Returns the minimum of two given float values.
116    * @param f1 Float input.
117    * @param f2 Float input.
118    * @return Minimum of f1 and f2.
119    */
120    public float min(float f1, float f2){
121        return (float)min((double)f1, (double)f2);
122    }
123 }

```

2 Einreichung mit syntaktischem Fehler:

```
1 public class LittleMath {
2     /**
3      * Returns absolute value of type long.
4      * @param l Long input
5      * @return Absolute value of l.
6      */
7     public long abs(long l){
8         if(l < 0 return -1;           //Error will occur!
9         return l;
10    }
11
12    /**
13     * Returns absolute value of type double.
14     * @param d Double input
15     * @return Absolute Value of d.
16     */
17    public double abs(double d){
18        if(d < 0) return -d;
19        return d;
20    }
21
22    /**
23     * Returns absolute value of type float.
24     * @param f Float input
25     * @return Absolute Value of f.
26     */
27    public float abs(float f){
28        return (float)abs((double)f);
29    }
30
31    /**
32     * Returns absolute value of type int.
33     * @param i Int input
34     * @return Absolute Value of i.
35     */
36    public int abs(int i){
37        return (int)abs((long)i);
38    }
39
40    /**
41     * Returns the maximum of two given long values.
42     * @param l1 Long input.
43     * @param l2 Long input.
44     * @return Maximum of l1 and l2.
45     */
46    public long max(long l1, long l2){
47        if (l1 < l2) return l2;
48        return l1;
49    }
50
51    /**
52     * Returns the maximum of two given int values.
53     * @param i1 Int input.
54     * @param i2 Int input.
55     * @return Maximum of i1 and i2.
56     */
57    public int max(int i1, int i2){
58        return (int)max((long)i1, (long)i2);
59    }
60
61    /**
62     * Returns the maximum of two given double values.
63     * @param d1 Double input.
```

```

64     * @param d2 Double input.
65     * @return Maximum of d1 and d2.
66     */
67     public double max(double d1, double d2){
68         if (d1 < d2) return d2;
69         return d1;
70     }
71
72     /**
73     * Returns the maximum of two given float values.
74     * @param f1 Float input.
75     * @param f2 Float input.
76     * @return Maximum of f1 and f2.
77     */
78     public double max(float f1, float f2){
79         return (float)max((double)f1, (double)f2);
80     }
81
82     /**
83     * Returns the minimum of two given long values.
84     * @param l1 Long input.
85     * @param l2 Long input.
86     * @return Minimum of l1 and l2.
87     */
88     public long min(long l1, long l2){
89         if (l1 < l2) return l1;
90         return l2;
91     }
92
93     /**
94     * Returns the minimum of two given int values.
95     * @param i1 Int input.
96     * @param i2 Int input.
97     * @return Minimum of i1 and i2.
98     */
99     public int min(int i1, int i2){
100        return (int)min((long)i1, (long)i2);
101    }
102
103    /**
104    * Returns the minimum of two given double values.
105    * @param d1 Double input.
106    * @param d2 Double input.
107    * @return Minimum of d1 and d2.
108    */
109    public double min(double d1, double d2){
110        if (d1 < d2) return d1;
111        return d2;
112    }
113
114    /**
115    * Returns the minimum of two given float values.
116    * @param f1 Float input.
117    * @param f2 Float input.
118    * @return Minimum of f1 and f2.
119    */
120    public float min(float f1, float f2){
121        return (float)min((double)f1, (double)f2);
122    }
123 }

```

3 Einreichung mit semantischem Fehler:

```
1 public class LittleMath {
2     /**
3      * Returns absolute value of type long.
4      * @param l Long input
5      * @return Absolute value of l.
6      */
7     public long abs(long l){
8         if(l < 0) return l;           //Semantically wrong
9         return l;
10    }
11
12    /**
13     * Returns absolute value of type double.
14     * @param d Double input
15     * @return Absolute Value of d.
16     */
17    public double abs(double d){
18        if(d < 0) return -d;
19        return d;
20    }
21
22    /**
23     * Returns absolute value of type float.
24     * @param f Float input
25     * @return Absolute Value of f.
26     */
27    public float abs(float f){
28        return (float)abs((double)f);
29    }
30
31    /**
32     * Returns absolute value of type int.
33     * @param i Int input
34     * @return Absolute Value of i.
35     */
36    public int abs(int i){
37        return (int)abs((long)i);
38    }
39
40    /**
41     * Returns the maximum of two given long values.
42     * @param l1 Long input.
43     * @param l2 Long input.
44     * @return Maximum of l1 and l2.
45     */
46    public long max(long l1, long l2){
47        if (l1 < l2) return l2;
48        return l1;
49    }
50
51    /**
52     * Returns the maximum of two given int values.
53     * @param i1 Int input.
54     * @param i2 Int input.
55     * @return Maximum of i1 and i2.
56     */
57    public int max(int i1, int i2){
58        return (int)max((long)i1, (long)i2);
59    }
60
61    /**
62     * Returns the maximum of two given double values.
63     * @param d1 Double input.
```

```

64     * @param d2 Double input.
65     * @return Maximum of d1 and d2.
66     */
67     public double max(double d1, double d2){
68         if (d1 < d2) return d2;
69         return d1;
70     }
71
72     /**
73     * Returns the maximum of two given float values.
74     * @param f1 Float input.
75     * @param f2 Float input.
76     * @return Maximum of f1 and f2.
77     */
78     public double max(float f1, float f2){
79         return (float)max((double)f1, (double)f2);
80     }
81
82     /**
83     * Returns the minimum of two given long values.
84     * @param l1 Long input.
85     * @param l2 Long input.
86     * @return Minimum of l1 and l2.
87     */
88     public long min(long l1, long l2){
89         if (l1 < l2) return l1;
90         return l2;
91     }
92
93     /**
94     * Returns the minimum of two given int values.
95     * @param i1 Int input.
96     * @param i2 Int input.
97     * @return Minimum of i1 and i2.
98     */
99     public int min(int i1, int i2){
100        return (int)min((long)i1, (long)i2);
101    }
102
103    /**
104    * Returns the minimum of two given double values.
105    * @param d1 Double input.
106    * @param d2 Double input.
107    * @return Minimum of d1 and d2.
108    */
109    public double min(double d1, double d2){
110        if (d1 < d2) return d1;
111        return d2;
112    }
113
114    /**
115    * Returns the minimum of two given float values.
116    * @param f1 Float input.
117    * @param f2 Float input.
118    * @return Minimum of f1 and f2.
119    */
120    public float min(float f1, float f2){
121        return (float)min((double)f1, (double)f2);
122    }
123 }

```


4 Unit-Tests:

```
1 private ${CLASS} c;
2
3 @Before public void setUp(){
4     c = new ${CLASS}();
5 }
6
7 @After public void tearDown(){
8     c = null;
9 }
10
11 @Test public void longAbs(){
12     assertEquals("abs(long) did not return the correct result.", 101, c.abs(101));
13     assertEquals("abs(long) did not return the correct result.", 101, c.abs(-101));
14 }
15
16 @Test public void intAbs(){
17     assertEquals("abs(int) did not return the correct result.", 99, c.abs(99));
18     assertEquals("abs(int) did not return the correct result.", 99, c.abs(-99));
19 }
20
21 @Test public void doubleAbs(){
22     assertEquals("abs(double) did not return the correct result.", 0.99d, c.abs(0.99d), 0d);
23     assertEquals("abs(double) did not return the correct result.", 0.99d, c.abs(-0.99d), 0d);
24 }
25
26 @Test public void floatAbs(){
27     assertEquals("abs(float) did not return the correct result.", 0.099f, c.abs(0.099f), 0f);
28     assertEquals("abs(float) did not return the correct result.", 0.099f, c.abs(-0.099f), 0f);
29 }
30
31 @Test public void longMax(){
32     long l1 = 99991;
33     long l2 = 11111;
34
35     assertEquals("max(long) did not return the correct result.", l1, c.max(l1, l2));
36     assertEquals("max(long) did not return the correct result.", l2, c.max(-l1, l2));
37 }
38
39 @Test public void intMax(){
40     int i1 = 234;
41     int i2 = 123;
42
43     assertEquals("max(int) did not return the correct result.", i1, c.max(i1, i2));
44     assertEquals("max(int) did not return the correct result.", i2, c.max(-i1, i2));
45 }
46
47 @Test public void doubleMax(){
48     double d1 = 0.234d;
49     double d2 = 0.000023d;
50
51     assertEquals("max(double) did not return the correct result.", d1, c.max(d1, d2));
52     assertEquals("max(double) did not return the correct result.", d2, c.max(-d1, d2));
53 }
54
55 @Test public void floatMax(){
56     float f1 = 0.34f;
57     float f2 = 0.11f;
58
59     assertEquals("max(float) did not return the correct result.", f1, c.max(f1, f2));
60     assertEquals("max(float) did not return the correct result.", f2, c.max(-f1, f2));
61 }
62
63 @Test public void longMin(){
```

```

64     long l1 = 99991;
65     long l2 = 11111;
66
67     assertEquals("min(long)_did_not_return_the_correct_result.", l2, c.min(l1, l2));
68     assertEquals("min(long)_did_not_return_the_correct_result.", -l1, c.min(-l1, l2));
69 }
70
71 @Test public void intMin(){
72     int i1 = 234;
73     int i2 = 123;
74
75     assertEquals("min(int)_did_not_return_the_correct_result.", i2, c.min(i1, i2));
76     assertEquals("min(int)_did_not_return_the_correct_result.", -i1, c.min(-i1, i2));
77 }
78
79 @Test public void doubleMin(){
80     double d1 = 0.234d;
81     double d2 = 0.000023d;
82
83     assertEquals("min(double)_did_not_return_the_correct_result.", d2, c.min(d1, d2));
84     assertEquals("min(double)_did_not_return_the_correct_result.", -d1, c.min(-d1, d2));
85 }
86
87 @Test public void floatMin(){
88     float f1 = 0.34f;
89     float f2 = 0.11f;
90
91     assertEquals("min(float)_did_not_return_the_correct_result.", f2, c.min(f1, f2));
92     assertEquals("min(float)_did_not_return_the_correct_result.", -f1, c.min(-f1, f2));
93 }

```