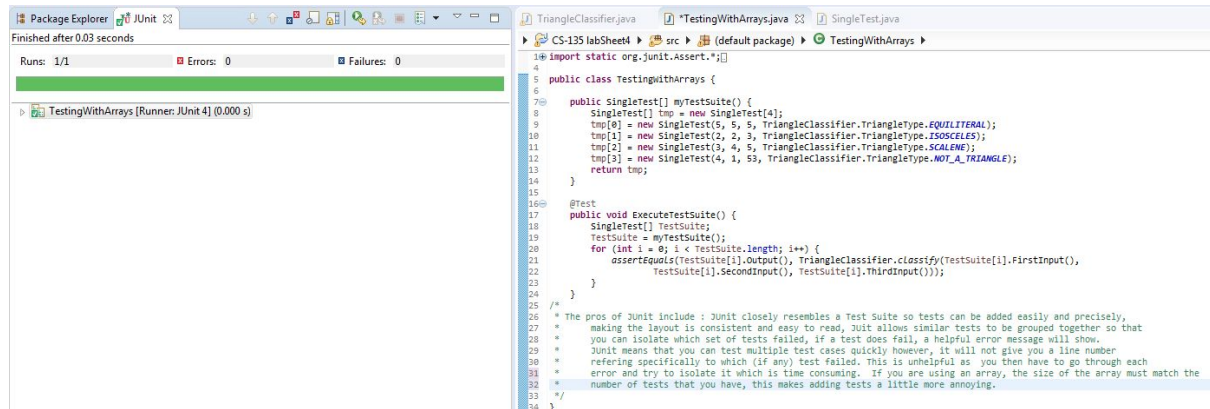


951428

960689

CS-135 Lab Sheet 4

Task 4.1



The screenshot shows an IDE with a JUnit test run on the left and the source code of `TestingWithArrays.java` on the right. The test run shows 1/1 runs, 0 errors, and 0 failures, with a duration of 0.000 s. The source code is as follows:

```
1 import static org.junit.Assert.*;
2
3
4
5 public class TestingWithArrays {
6
7     public SingleTest[] myTestSuite() {
8         SingleTest[] tmp = new SingleTest[4];
9         tmp[0] = new SingleTest(5, 5, 5, TriangleClassifier.TriangleType.EQUILATERAL);
10        tmp[1] = new SingleTest(2, 2, 3, TriangleClassifier.TriangleType.ISOSCELES);
11        tmp[2] = new SingleTest(3, 4, 5, TriangleClassifier.TriangleType.SCALENE);
12        tmp[3] = new SingleTest(4, 1, 53, TriangleClassifier.TriangleType.NOT_A_TRIANGLE);
13        return tmp;
14    }
15
16    @Test
17    public void ExecuteTestSuite() {
18        SingleTest[] TestSuite;
19        TestSuite = myTestSuite();
20        for (int i = 0; i < TestSuite.length; i++) {
21            assertEquals(TestSuite[i].Output(), TriangleClassifier.classify(TestSuite[i].FirstInput(),
22                                TestSuite[i].SecondInput(), TestSuite[i].ThirdInput()));
23        }
24    }
25
26    /* The pros of JUnit include: JUnit closely resembles a Test Suite so tests can be added easily and precisely,
27     * making the layout is consistent and easy to read, JUnit allows similar tests to be grouped together so that
28     * you can isolate which set of tests failed, if a test does fail, a helpful error message will show.
29     * JUnit means that you can test multiple test cases quickly however, it will not give you a line number
30     * referring specifically to which (if any) test failed. This is unhelpful as you then have to go through each
31     * error and try to isolate it which is time consuming. If you are using an array, the size of the array must match the
32     * number of tests that you have, this makes adding tests a little more annoying.
33     */
34 }
```

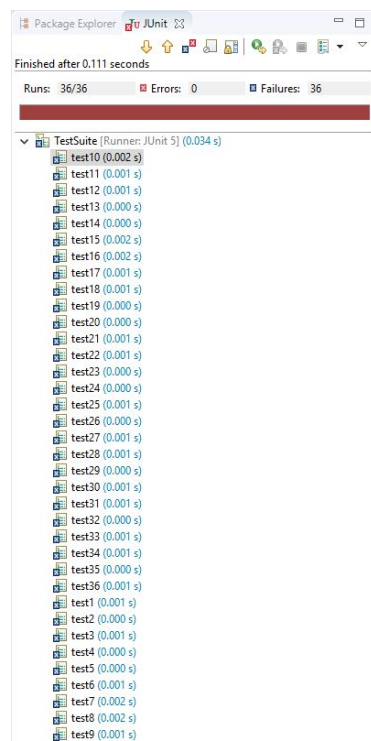
Task 4.2

```
1 import static org.junit.Assert.*;
2
3
4
5 public class TestSuite {
6
7     @Test
8     public void test1() {
9         assertEquals("15.6.---2000", nextDateFunction.tomorrow(6, 14, 2000));
10    }
11
12    @Test
13    public void test2() {
14        assertEquals("15.6.---1996", nextDateFunction.tomorrow(6, 14, 1996));
15    }
16
17    @Test
18    public void test3() {
19        assertEquals("15.6.---2002", nextDateFunction.tomorrow(6, 14, 2002));
20    }
21
22
23    @Test
24    public void test4() {
25        assertEquals("30.6.---2000", nextDateFunction.tomorrow(6, 29, 2000));
26    }
27
28    @Test
29    public void test5() {
30        assertEquals("30.6.---1996", nextDateFunction.tomorrow(6, 29, 1996));
31    }
32
33    @Test
34    public void test6() {
35        assertEquals("30.6.---2002", nextDateFunction.tomorrow(6, 29, 2002));
36    }
37
38
39    @Test
40    public void test7() {
41        assertEquals("1.7.---2000", nextDateFunction.tomorrow(6, 30, 2000));
42    }
43
44    @Test
45    public void test8() {
46        assertEquals("1.7.---1996", nextDateFunction.tomorrow(6, 30, 1996));
47    }
48
49    @Test
50    public void test9() {
51        assertEquals("1.7.---2002", nextDateFunction.tomorrow(6, 30, 2002));
52    }
53
54
55    @Test
56    public void test10() {
57        assertEquals("day out of range", nextDateFunction.tomorrow(6, 31, 2000));
58    }
59
60    @Test
61    public void test11() {
62        assertEquals("day out of range", nextDateFunction.tomorrow(6, 31, 1996));
63    }
64
65    @Test
66    public void test12() {
67        assertEquals("day out of range", nextDateFunction.tomorrow(6, 31, 2002));
68    }
69
70
71    @Test
72    public void test13() {
73        assertEquals("15.7.---2000", nextDateFunction.tomorrow(7, 14, 2000));
74    }
75
76    @Test
77    public void test14() {
78        assertEquals("15.7.---1996", nextDateFunction.tomorrow(7, 14, 1996));
79    }
80
81    @Test
82    public void test15() {
83        assertEquals("15.7.---2002", nextDateFunction.tomorrow(7, 14, 2002));
84    }
85
86
87    @Test
88    public void test16() {
89        assertEquals("30.7.---2000", nextDateFunction.tomorrow(7, 29, 2000));
90    }
91
92    @Test
93    public void test17() {
94        assertEquals("30.7.---1996", nextDateFunction.tomorrow(7, 29, 1996));
95    }
96
97    @Test
98    public void test18() {
99        assertEquals("30.7.---2002", nextDateFunction.tomorrow(7, 29, 2002));
100   }
101
102
103    @Test
104    public void test19() {
105        assertEquals("31.7.---2000", nextDateFunction.tomorrow(7, 30, 2000));
106    }
107
108 }
```

```

108 @Test
109 public void test20() {
110     assertEquals("31.7.---1996", nextDateFunction.tomorrow(7, 30, 1996));
111 }
112
113 @Test
114 public void test21() {
115     assertEquals("31.7.---2002", nextDateFunction.tomorrow(7, 30, 2002));
116 }
117
118
119 @Test
120 public void test22() {
121     assertEquals("1.8.---2000", nextDateFunction.tomorrow(7, 31, 2000));
122 }
123
124 @Test
125 public void test23() {
126     assertEquals("1.8.---1996", nextDateFunction.tomorrow(7, 31, 1996));
127 }
128
129 @Test
130 public void test24() {
131     assertEquals("1.8.---2002", nextDateFunction.tomorrow(7, 31, 2002));
132 }
133
134
135 @Test
136 public void test25() {
137     assertEquals("15.2.---2000", nextDateFunction.tomorrow(2, 14, 2000));
138 }
139
140 @Test
141 public void test26() {
142     assertEquals("15.2.---1996", nextDateFunction.tomorrow(2, 14, 1996));
143 }
144
145 @Test
146 public void test27() {
147     assertEquals("15.2.---2002", nextDateFunction.tomorrow(2, 14, 2002));
148 }
149
150
151 @Test
152 public void test28() {
153     assertEquals("1.3.---2000", nextDateFunction.tomorrow(2, 29, 2000));
154 }
155
156 @Test
157 public void test29() {
158     assertEquals("1.3.---1996", nextDateFunction.tomorrow(2, 29, 1996));
159 }
160
161 @Test
162 public void test30() {
163     assertEquals("day out of range", nextDateFunction.tomorrow(2, 29, 2002));
164 }
165
166
167 @Test
168 public void test31() {
169     assertEquals("day out of range", nextDateFunction.tomorrow(2, 30, 2000));
170 }
171
172 @Test
173 public void test32() {
174     assertEquals("day out of range", nextDateFunction.tomorrow(2, 30, 1996));
175 }
176
177 @Test
178 public void test33() {
179     assertEquals("day out of range", nextDateFunction.tomorrow(2, 30, 2002));
180 }
181
182
183 @Test
184 public void test34() {
185     assertEquals("day out of range", nextDateFunction.tomorrow(2, 31, 2000));
186 }
187
188 @Test
189 public void test35() {
190     assertEquals("day out of range", nextDateFunction.tomorrow(2, 31, 1996));
191 }
192
193 @Test
194 public void test36() {
195     assertEquals("day out of range", nextDateFunction.tomorrow(2, 31, 2002));
196 }
197

```



Above are images documenting every test case encoded as separate methods as well as an image showing the JUnit results. Comparing each test method to its test in the table will prove that the input and expected output have been completed for every test and that the right method from nextDayFunction is being tested.