```java
import static org.hamcrest.CoreMatchers.equalTo;

public class BruchTest {

    Bruch einBruch = new Bruch();
    Bruch einBruch2 = new Bruch(7);
    Bruch einBruch3 = new Bruch(7,7);
    Bruch einBruch4 = new Bruch(3,4);
    Bruch einBruch5 = new Bruch(4,3);
    Bruch einBruch6 = new Bruch(2,4);

    Bruch b1PlusB2 = einBruch.add(einBruch2);
    Bruch b3PlusB4 = einBruch3.add(einBruch4);
    Bruch b4PlusB5 = einBruch4.add(einBruch5);

    Bruch b1SubB2 = einBruch.sub(einBruch2);
    Bruch b3SubB4 = einBruch3.sub(einBruch4);
    Bruch b4SubB5 = einBruch4.sub(einBruch5);
    Bruch b4SubB6 = einBruch4.sub(einBruch6);

    Bruch b1MulB2 = einBruch .mul(einBruch2);
    Bruch b3MulB4 = einBruch3.mul(einBruch4);
    Bruch b4MulB5 = einBruch4.mul(einBruch5);
    Bruch b4MulB6 = einBruch4.mul(einBruch6);

    Bruch b1DivB2 = einBruch .div(einBruch2);
    Bruch b3DivB4 = einBruch3.div(einBruch4);
    Bruch b4DivB5 = einBruch4.div(einBruch5);
    Bruch b4DivB6 = einBruch4.div(einBruch6);

    /* Ein beispielhafter Test einer Ausnahmebehandlung: */
    @Test
    public void testKonstruktor() {

        // Teste das Ausloesen einer Ausnahme bei Null als Nenner. Da der Konstruktor setNenner aufruhft muss setNenner nichtmehr getestet werden.
        Throwable ex = assertThrows(IllegalArgumentException.class, () -> {new Bruch(42, 0);});
        assertThat("Inkorrekte Ausnahme geworfen.",
                ex.getMessage(),
                is(equalTo("Nenner muss von Null verschieden sein.")));
        assertThat("Konstruktor arbeitet nicht korrekt.",
                einBruch,
                is(equalTo(new Bruch(1,1))));
        assertThat("Konstruktor arbeitet nicht korrekt.",
                einBruch2,
                is(equalTo(new Bruch(7, 1))));
        assertThat("Konstruktor arbeitet nicht korrekt.",
                einBruch3,
                is(equalTo(einBruch)));
    }

    /* Ein beispielhafter Test: */
```

```
57      @Test
58      public void testGetZaehler() {
59          // Erzeuge einen Bruch zum Testen.
60          // Teste den Zaehler des Bruchs.
61          assertThat("getZahler arbeitet nicht korrekt.",
62                  einBruch.getZaehler(),
63                  is(equalTo(1)));
64          assertThat("getZahler arbeitet nicht korrekt.",
65                  einBruch2.getZaehler(),
66                  is(equalTo(7)));
67          assertThat("kuerze arbeitet nicht korrekt",
68                  einBruch3.getZaehler(),
69                  is(equalTo(1)));
70          assertThat("getZahler arbeitet nicht korrekt.",
71                  einBruch4.getZaehler(),
72                  is(equalTo(3)));
73          assertThat("getNenner arbeitet nicht korrekt.",
74                  einBruch4.getZaehler(),
75                  is(equalTo(3)));
76
77      }
78      @Test
79      public void testGetNenner() {
80          // Erzeuge einen Bruch zum Testen.
81
82          // Teste den Zaehler des Bruchs.
83          assertThat("getZahler arbeitet nicht korrekt.",
84                  einBruch.getNenner(),
85                  is(equalTo(1)));
86
87      }
88
89      @Test
90      public void testAdd() {
91          // Erzeuge einen Bruch zum Testen.
92
93
94          // Teste den Zaehler des Bruchs.
95          assertThat("add arbeitet nicht korrekt.",
96                  b1PlusB2,
97                  is(equalTo(new Bruch(8, 1))));
98          assertThat("add arbeitet nicht korrekt.",
99                  b3PlusB4,
100                 is(equalTo(new Bruch(7, 4))));
101         assertThat("add arbeitet nicht korrekt.",
102                 b4PlusB5,
103                 is(equalTo(new Bruch(25, 12))));
104
105     }
106     @Test
107     public void testSub() {
108         // Erzeuge einen Bruch zum Testen.
109
```

```
110
111         // Teste den Zaehler des Bruchs.
112         assertThat("sub arbeitet nicht korrekt.",
113                 b1SubB2,
114                 is(equalTo(new Bruch(-6, 1))));
115         assertThat("sub arbeitet nicht korrekt.",
116                 b3SubB4,
117                 is(equalTo(new Bruch(1, 4))));
118         assertThat("sub arbeitet nicht korrekt.",
119                 b4SubB5,
120                 is(equalTo(new Bruch(-7, 12))));
121         assertThat("sub arbeitet nicht korrekt.",
122                 b4SubB6,
123                 is(equalTo(new Bruch(1, 4))));
124
125     }
126
127     @Test
128     public void testMul() {
129         // Erzeuge einen Bruch zum Testen.
130
131
132         // Teste den Zaehler des Bruchs.
133         assertThat("mul arbeitet nicht korrekt.",
134                 b1MulB2,
135                 is(equalTo(new Bruch(7, 1))));
136         assertThat("mul arbeitet nicht korrekt.",
137                 b3MulB4,
138                 is(equalTo(new Bruch(3, 4))));
139         assertThat("mul arbeitet nicht korrekt.",
140                 b4MulB5,
141                 is(equalTo(new Bruch(1, 1))));
142         assertThat("mul arbeitet nicht korrekt.",
143                 b4MulB6,
144                 is(equalTo(new Bruch(3, 8))));
145
146     }
147
148     @Test
149     public void testDiv() {
150         // Erzeuge einen Bruch zum Testen.
151
152
153         // Teste den Zaehler des Bruchs.
154         assertThat("mul arbeitet nicht korrekt.",
155                 b1DivB2,
156                 is(equalTo(new Bruch(1, 7))));
157         assertThat("mul arbeitet nicht korrekt.",
158                 b3DivB4,
159                 is(equalTo(new Bruch(4, 3))));
160         assertThat("mul arbeitet nicht korrekt.",
161                 b4DivB5,
162                 is(equalTo(new Bruch(9, 16))));
```

```
163         assertThat("mul arbeitet nicht korrekt.",
164                 b4DivB6,
165                 is(equalTo(new Bruch(3, 2))));
166 
167     }
168 }
169 
```