

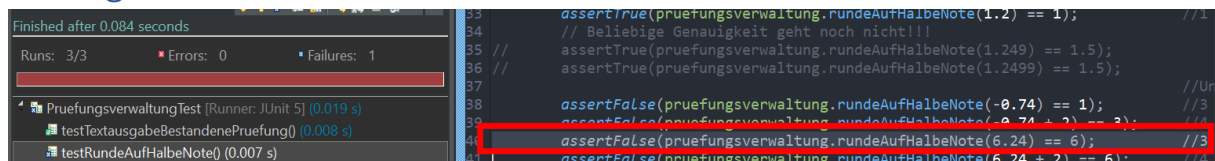
1. Hinzufügen neuer Äquivalenzklassen, inkl. Negativer:

```
@Test
void testRundeAufHalbeNote() {
    //Gültige Äquivalenzklassen:
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.24) == 1); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.24 + 1) == 1); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.74) == 1.5); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.74 + 1) == 1.5); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.25) == 1.5); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.25 + 1) == 1.5); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.75) == 2); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.75 + 1) == 2); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0) == 0); //1

    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1) == 1); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.2) == 1); //1
    // Beliebige Genauigkeit geht noch nicht!!!
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.249) == 1.5);
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.2499) == 1.5);

    //Ungültige Äquivalenzklassen:
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(-0.24) == 1); //3
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(-0.74 - 2) == 1); //4
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(6.24) == 6); //5
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(6.24 + 1) == 6); //6
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(-3.75 + 4) == 1); //7
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(8.24 - 2) == 6); //8
}
```

2. Dabei wurde ein Fehler entdeckt, die Note 6.24 dürfe es nicht geben. Laut Testresultat wird diese aber (anscheinend) auf 6 gesetzt:



3. Einfügen der Änderung: Es wird nun davon ausgegangen, dass nur «gültige Noten», zwischen 1 und 6 gerundet werden. Ist eine Note ungültig, wird sie auf 1 gesetzt (durchgefallen):

```
double rundeAufHalbeNote(double note) {
    if (note <= 6.0 && note >= 1.0) {
        return Math.round(note * 2) / 2.0;
    } else {
        return note = 1;
    }
}
```

NEU

4. Anpassung der Tests:

1. PruefungsverwaltungTest.java

```
@Test
void testRundeAufHalbeNote() {
    //Gültige Äquivalenzklassen:
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.24) == 1); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.24 + 1) == 1); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.74) == 1.5); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.74 + 1) == 1.5); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.25) == 1.5); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.25 + 1) == 1.5); //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.75) == 2); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.75 + 1) == 2); //2
    assertNull(pruefungsverwaltung.rundeAufHalbeNote(0) == 0); //1

    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1) == 1); //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.2) == 1); //1
    // Beliebige Genauigkeit geht noch nicht!!!
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.249) == 1.5);
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.2499) == 1.5);

    //Ungültige Äquivalenzklassen:
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(-0.24) == 1); //3
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(-0.74 - 2) == 1); //4
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(6.24) == 6); //5
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(6.24 + 1) == 6); //6
    assertNotEquals(1, pruefungsverwaltung.rundeAufHalbeNote(-3.75 + 4)); //7
    assertFalse(pruefungsverwaltung.rundeAufHalbeNote(8.24 - 2) == 6); //8
}
```

2. ZufaeiligeNotengebungTest.java

```
ZufaeiligeNotengebungTest.java ✕
4 • @AfterEach
5   void tearDown() throws Exception {
6   }
7
8 • @Test
9   void testGeneriereZufaeiligePruefungsnote() {
10     for (int i = 0; i < 1000000; i++) {
11         assertTrue(zufaeiligeNotengebung.generiereZufaeiligePruefungsnote() < 6.0);
12         assertTrue(zufaeiligeNotengebung.generiereZufaeiligePruefungsnote() > 1.0);
13     }
14 }
15 }
```

5. Testresultate

Finished after 0.092 seconds

Runs: 3/3 ✕ Errors: 0 * Failures: 0

4 PruefungsverwaltungTest [Runner: JUnit 5] (0.021 s)

- testTextausgabeBestandenePruefung() (0.008 s)
- testRundeAufHalbeNote() (0.006 s)
- testTextausgabeNichtBestandenePruefung() (0.006 s)