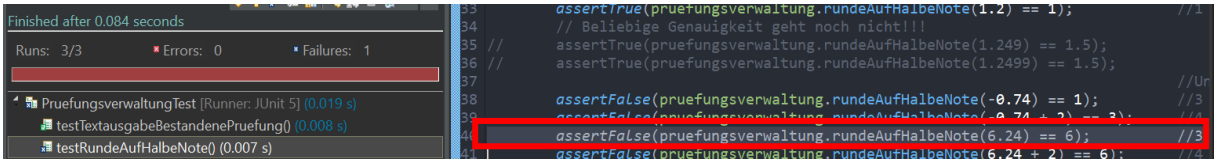
1. Hinzufügen neuer Äquivalenzklassen, inkl. Negativer:

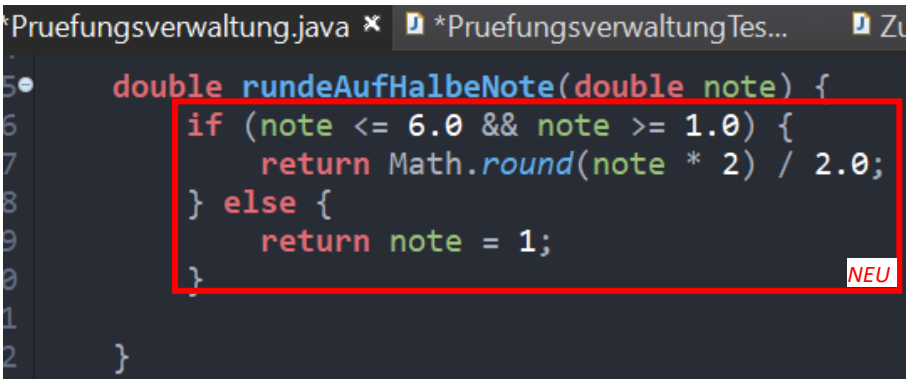
```java
@Test
void testRundeAufHalbeNote() {                                  //Gültige Äquivalenzklassen:
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.24) == 1);          //1
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(0.24 + 1) == 1);      //2
    assertTrue(pruefungsverwaltung.rundeAufHalbeNote(1.74) == 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
    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):



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
    assertNotNull(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
    assertFalse(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. ZufaelligeNotengebungTest.java

ZufaelligeNotengebungTest.java

```
    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void testGeneriereZufaelligePruefungsnote() {
        for (int i = 0; i < 1000000; i++) {
            assertTrue(zufaelligeNotengebung.generiereZufaelligePruefungsnote() < 6.0);
            assertTrue(zufaelligeNotengebung.generiereZufaelligePruefungsnote() > 1.0);
        }
    }
}
```

5. Testresultate