Florian Jäger

Dreieckstests

18.04.2016

Aufgabenstellung

Die Schüler sollen mit dem Eclipseplugin EclEmma und JUnit eine Klasse vollständig zu testen und mit EclEmma die Klasse zu 100 Prozent covern. Man soll für jede Methode der Klasse Dreieck eine Testklasse erstellen. Weiters soll man nach jeder erstellten Testmethode commiten und diese auf sein Githubrepository hochladen.

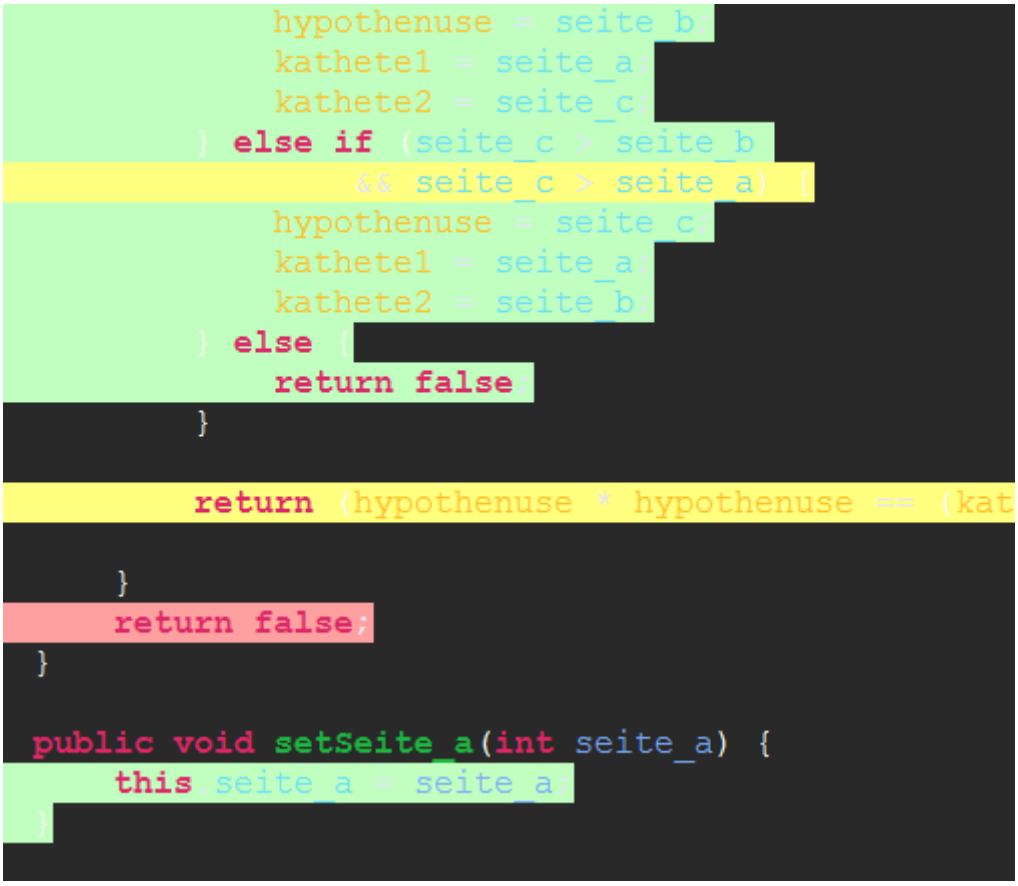
Allgemeine Erklärung zu EclEmma

1. Im Eclipse Marketplace EclEmma herunterladen
2. Installieren

EclEmma fügt einen neuen Runbutton hinzu, der die Coverage der zu testenden Klasse zeigt.
Mit Rot, Gelb und Rot wird signalisiert, welche Zeilen die in der CUT gecovered wurden.

Rot = nichts in der Zeile gecovered
Gelb = Teile in der Zeile gecovered
Grün = alles in der Zeile gecovered

Anzeige der Coverage:



Florian Jäger

Dreieckstests

18.04.2016

TestIstDreieck

Die zu testende Methode:

```
public boolean istDreieck() {
    if (this.seite_a == 0
            || this.seite_b == 0
            || this.seite_c == 0) {
        return false;
    }
    if (this.seite_a < 0
            || this.seite_b < 0
            || this.seite_c < 0) {
        return false;
    }
    if ((this.seite_a + this.seite_b == this.seite_c)
            || (this.seite_a + this.seite_c == this.seite_b)
            || (this.seite_c + this.seite_b == this.seite_a)) {
        return false;
    }
    if (this.seite_a + this.seite_b < 0
            || this.seite_b + this.seite_c < 0
            || this.seite_a + this.seite_c < 0) {
        return false;
    }
    if ((this.seite_a + this.seite_b < this.seite_c)
            || (this.seite_a + this.seite_c < this.seite_b)
            || (this.seite_c + this.seite_b < this.seite_a)) {
        return false;
    }

    return true;
}
```

```
@Test
public void testIstDreieckANull() {
    dreieck.setSeite_a(0);
    dreieck.setSeite_b(1);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}

@Test
public void testIstDreieckBNull() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(0);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}

@Test
public void testIstDreieckCNull() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(0);

    assertFalse(dreieck.istDreieck());
}
```

TestABCnegativ:

Es wird getestet, ob eine der drei Seiten Null ist.

Florian Jäger

Dreieckstests

18.04.2016

```
@Test
public void testIstDreieckANegativ() {
    dreieck.setSeite_a(-1);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(3);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckBNegativ() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(-2);
    dreieck.setSeite_c(3);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckCNegativ() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(-3);

    assertFalse(dreieck.istDreieck());
}
```

TestABCNegativ:

Es wird getestet, ob eine der drei Seiten negativ ist.

```
@Test
public void testIstDreieckAplusBistgleichC() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(3);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckAplusCistgleichB() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(3);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckBplusCistgleichA() {
    dreieck.setSeite_a(3);
    dreieck.setSeite_b(1);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}
```

TestABCplusABCgleichABC:

Es wird getestet, ob:

A+B=C
A+C=B
B+C=A

Florian Jäger

Dreieckstests

18.04.2016

```
@Test
public void testIstDreieckAplusBkleinerNull() {
    dreieck.setSeite_a(Integer.MAX_VALUE);
    dreieck.setSeite_b(3);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckBplusCkleinerNull() {
    dreieck.setSeite_a(3);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(Integer.MAX_VALUE);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckAplusCkleinerNull() {
    dreieck.setSeite_a(Integer.MAX_VALUE-10);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(11);

    assertFalse(dreieck.istDreieck());
}
```

TestABCplusABCkleinerNull:

Es wird getestet, ob:

A+B<0
A+C<0
B+C<0

```
@Test
public void testIstDreieckAplusBkleinerC() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(4);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckAplusCkleinerB() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(4);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}
@Test
public void testIstDreieckBplusCkleinerA() {
    dreieck.setSeite_a(4);
    dreieck.setSeite_b(1);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.istDreieck());
}
```

TestABCplusABCkleinerABC:

Es wird getestet, ob:

A+B<C
A+C<B
B+C<A

Florian Jäger

Dreieckstests

18.04.2016

TestGleichseitig

Die zu testende Methode:

```
public boolean gleichSeitig() {
    return this.istDreieck() && (this.seite_a == this.seite_b && this.seite_b == this.seite_c);
}
```

```
@Test
public void testGleichsteitignotA() {
    dreieck.setSeite_a(1);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.gleichSeitig());
}
@Test
public void testGleichsteitignotB() {
    dreieck.setSeite_a(4);
    dreieck.setSeite_b(2);
    dreieck.setSeite_c(4);

    assertFalse(dreieck.gleichSeitig());
}
@Test
public void testGleichsteitignotC() {
    dreieck.setSeite_a(4);
    dreieck.setSeite_b(4);
    dreieck.setSeite_c(2);

    assertFalse(dreieck.gleichSeitig());
```

TestGleichseitig!ABC:

Es wird getestet, ob eine der Seiten nicht gleich seitig ist.

Florian Jäger

Dreieckstests

18.04.2016

TestGleichschenkelig

Die zu testende Methode:

```
public boolean gleichSchenkelig() {
    return this.istDreieck()
            && (this.seite_a == this.seite_b
                || this.seite_b == this.seite_c
                || this.seite_a == this.seite_c);
}
```

```
@Test
public void testGleichschenkeligAgleichB() {
    dreieck.setSeite_a(4);
    dreieck.setSeite_b(4);
    dreieck.setSeite_c(5);

    assertTrue(dreieck.gleichSchenkelig());
}
@Test
public void testGleichschenkeligBgleichC() {
    dreieck.setSeite_a(4);
    dreieck.setSeite_b(5);
    dreieck.setSeite_c(5);

    assertTrue(dreieck.gleichSchenkelig());
}
@Test
public void testGleichschenkeligAgleichC() {
    dreieck.setSeite_a(5);
    dreieck.setSeite_b(8);
    dreieck.setSeite_c(5);

    assertTrue(dreieck.gleichSchenkelig());
}
```

TestGleichschenkeligABCgleichABC:

Es wird getestet, ob zwei der drei Seiten gleichschenkelig sind.

Florian Jäger

Dreieckstests

18.04.2016

TestRechtwinkelig

Die zu testende Methode:

```
public boolean rechtWinkelig() {
    if (this.istDreieck()) {

        int hypothenuse = 0;
        int kathete1 = 0;
        int kathete2 = 0;

        if (seite_a > seite_b
                && seite_a > seite_c) {
            hypothenuse = seite_a;
            kathete1 = seite_b;
            kathete2 = seite_c;
        } else if (seite_b > seite_a
                && seite_b > seite_c) {
            hypothenuse = seite_b;
            kathete1 = seite_a;
            kathete2 = seite_c;
        } else if (seite_c > seite_b
                && seite_c > seite_a) {
            hypothenuse = seite_c;
            kathete1 = seite_a;
            kathete2 = seite_b;
        } else {
            return false;
        }

        return (hypothenuse * hypothenuse == (kathete1 * kathete1 + kathete2 * kathete2));

    }
    return false;
}
```

```
@Test
public void testRechtWinkeligAgroeßerBundAgroeßerC() {
    dreieck.setSeite_a(80);
    dreieck.setSeite_b(60);
    dreieck.setSeite_c(40);

    assertFalse(dreieck.rechtWinkelig());
}
@Test
public void testRechtWinkeligBgroeßerAundBGroeßerC() {
    dreieck.setSeite_a(60);
    dreieck.setSeite_b(80);
    dreieck.setSeite_c(40);
    assertFalse(dreieck.rechtWinkelig());
}
@Test
public void testRechtWinkeligCgroeßerBundCgroeßerA() {
    dreieck.setSeite_a(40);
    dreieck.setSeite_b(60);
    dreieck.setSeite_c(80);

    assertFalse(dreieck.rechtWinkelig());
}
```

TestRechtwinkeligABCgroeßerA
Es wird getestet, welche Seite die Längste ist.

Florian Jäger

Dreieckstests

18.04.2016

Testsuite

Ein Testsuite wird bei dieser Aufgabe benötigt, da wir mehrere Testklassen haben und wir sie alle gleichzeitig ausführen müssen.

```
package dreieck;

import org.junit.runner.RunWith;

@RunWith(Suite.class)
@SuiteClasses({ TestIstDreieck.class, TestGleichseitig.class, TestGleichschenkelig.class, TestRechtwinkelig.class })
public class Testsuite {

}
```

Florian Jäger

Dreieckstests

18.04.2016

GIT

Ein Repository erstellen in dem Projektorder:

```
Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git init
Initialized empty Git repository in C:/Users/Florian/Desktop/3AHIT/SEW/CoverageU
ebung/.git/
```

Das Remoterepository pullen:

```
Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git pull origin master
From https://github.com/fjaeger-tgm/SEW_15-16
 * branch            master     -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 .gitignore                          |  12 +
 LICENSE                             | 674 ++++++++++++++++++++++++++++++++++++++
 README.md                           |   3 +
 "j\303\244ger/TestStringTokenizer.java" |  41 ++
 4 files changed, 730 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
 create mode 100644 README.md
 create mode 100644 "j\303\244ger/TestStringTokenizer.java"
```

Die Projektorder adden und nach jeder Methode commiten:

```
Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git add src

Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git add test

Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   src/dreieck/Dreieck.java
        new file:   test/dreieck/TestIstDreieck.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .classpath
        .project
        .settings/
        bin/


Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git commit -m "Added Dreieck and TestIstDreieck"
[master (root-commit) 8aa2730] Added Dreieck and TestIstDreieck
 2 files changed, 217 insertions(+)
 create mode 100644 src/dreieck/Dreieck.java
 create mode 100644 test/dreieck/TestIstDreieck.java
```

Pushen:

```
Florian@Laptop MINGW64 ~/Desktop/3AHIT/SEW/CoverageUebung (master)
$ git push --set-upstream origin master
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 2.02 KiB | 0 bytes/s, done.
Total 10 (delta 1), reused 0 (delta 0)
To https://github.com/fjaeger-tgm/SEW_15-16
   199b2a2..456459b  master -> master
Branch master set up to track remote branch master from origin.
```