Florian Wellner 3AHIT 19.04.2016

100% Coverage für die Klasse Dreieck.java ist das Ziel, für jede Methode die zu covern ist, wird eine neue Testklasse erstellt.
Ein lokales Reposetory erstellen (optional neues online Reposetory erstellen), remote adden, upstream setten.

1. Testklasse erstellen: **TestIstDreieck.java**

Um die Methoden überprüfen zu können, wäre es eine gute Idee Objekte der Klasse Dreieck.java zu erstellen:

private Dreieck dreieck;
in setUp() initialisieren: this.dreieck = new Dreieck(seite_a, seite_b, seite_c);

Die erste Methode **istDreieck()** wird von 16 Methoden (Testfälllen) gecovered:

istDreieckSeiteANull()...covered den Fall, dass seite_a 0 ist.

assertFalse(dreieck.istDreieck()); **WICHTIG!!!** assertFalse(), weil es KEIN Dreieck ist, es erfüllt nicht die Voraussetzungen um ein Dreieck zu sein.

istDreieckSeiteBNull()...covered den Fall, dass seite_b 0 ist.

istDreieckSeiteCNull()...covered den Fall, dass seite_c 0 ist.

istDreieckSeiteAKleinerNull()...covered den Fall, dass seite_a kleiner 0 ist.

istDreieckSeiteBKleinerNull()...covered den Fall, dass seite_b kleiner 0 ist.

istDreieckSeiteCKleinerNull()...covered den Fall, dass seite_c kleiner 0 ist.

istDreieckSeiteAPlusSeiteBGleichC()...covered den Fall, dass seite_a + seite_b = seite_c ist.

istDreieckSeiteAPlusSeiteCGleichB()...covered den Fall, dass seite_a + seite_c = seite_b ist.

istDreieckSeiteCPlusSeiteBGleichA()...covered den Fall, dass seite_c + seite_b = seite_a ist.

istDreieckSeiteAPlusSeiteBKleinerNull()...covered den Fall, dass seite_a + seite_b < 0

Um zu erreichen, dass seite_a + seite_b < 0 ist, ist die statische Konstante **MAX_VALUE** zu benutzen, **MAX_VALUE** + 1 = eine negative Zahl, da nach dem maximalen Wert der minimale Wert kommt. Da aber **MAX_VALUE** + 1 nicht als negative Zahl angesehen ist bei der Coverage (aus welchem Grund auch immer) wird hierfür einfach **MAX VALUE** -1 +2 verwendet.

istDreieckSeiteBPlusSeiteCKleinerNull()...covered den Fall, dass seite_b + seite_c < 0

istDreieckSeiteAPlusSeiteCKleinerNull()...covered den Fall, dass seite_a + seite_c < 0

istDreieckSeiteAPlusSeiteBKleinerC()...covered den Fall, dass seite_a + seite_b < seite_c ist.

istDreieckSeiteAPlusSeiteCKleinerB()...covered den Fall, dass seite_a + seite_c < seite_b ist.

istDreieckSeiteCPlusSeiteBKleinerA()...covered den Fall, dass seite_c + seite_b < seite_a ist.

istDreieckTrue()...covered den Fall, dass es sich wirklich um ein Dreieck handelt, mit assertTrue();

git commit -m“text“ und git push. Nach jeder fertiggestellten Testklasse pushen.

2. Testklasse erstellen: **TestIstDreieckGleichseitig.java**

private Dreieck dreieck;
in setUp() initialisieren: this.dreieck = new Dreieck(seite_a, seite_b, seite_c);

Die Methode **gleichSeitig()** wird von sechs Methoden (Testfälllen) gecovered:

istNurDreieckTrue()...covered den Fall, dass es sich um ein allgemeines Dreieck handelt.

istNurDreieckFalse()...covered den Fall, dass es sich um kein Dreieck handelt.

istABFalseUndBCFalse()...covered den Fall, dass a =/= b und b =/= c.

istABFalseUndBCTrue()...covered den Fall, dass a =/= b und b = c.

istABTrueUndBCFalse()...covered den Fall, dass a = b und b =/= c.

istABTrueUndBCTrue()...covered den Fall, dass a = b und b = c, es ist gleichseitig -> **assertTrue()**

git commit -m“text“ und git push.

3. Testklasse erstellen: **TestIstDreieckGleichschenkelig.java**

private Dreieck dreieck;
in setUp() initialisieren: this.dreieck = new Dreieck(seite_a, seite_b, seite_c);

Die Methode **gleichSchenkelig()** wird von fünf Methoden (Testfälllen) gecovered:

istNurDreieckTrue()...covered den Fall, dass es sich um ein allgemeines Dreieck handelt.

istNurDreieckFalse()...covered den Fall, dass es sich um kein Dreieck handelt.

istDreieckTrueUndAGleichB()...covered den Fall, dass es ein Dreieck ist und dass a = b ist. assertTrue()

istDreieckTrueUndBGleichC()...covered den Fall, dass es ein Dreieck ist und dass b = c ist. assertTrue()

istDreieckTrueUndAGleichC()...covered den Fall, dass es ein Dreieck ist und dass a = c ist. assertTrue()

git commit -m“text“ und git push.

4. Testklasse erstellen: **TestIstDreieckRechtWinkelig.java**

private Dreieck dreieck;
in setUp() initialisieren: this.dreieck = new Dreieck(seite_a, seite_b, seite_c);

Die Methode **rechtWinkelig()** wird von acht Methoden (Testfälllen) gecovered:

istNurDreieckTrue()...covered den Fall, dass es sich um ein allgemeines Dreieck handelt.

istNurDreieckFalse()...covered den Fall, dass es sich um kein Dreieck handelt.

IstABFalseUndACFalse()...covered den Fall, dass a < b und a < c ist.

IstABFalseUndACTrue()...covered den Fall, dass a < b und a > c ist.

IstABTrueUndACFalse()...covered den Fall, dass a > b und a < c ist.

IstABTrueUndACTrue()...covered den Fall, dass a > b und a > c ist. assertTrue()

istBAFalseUndBCFalse()...covered den Fall, dass b < a und b < c ist.

IstCBFalseUndCAFalse()...covered den Fall, dass c < b und c < a ist.

git commit -m“text“ und git push.

Letzte Klasse erstellen, **AllTests**, die alle Testklassen zusammenfasst, eine sogenannte **Suiteclass**, um die Klasse **Dreieck.java** komplett mit allen Testklassen 100% zu covern.

git commit -m“text“ und git push.