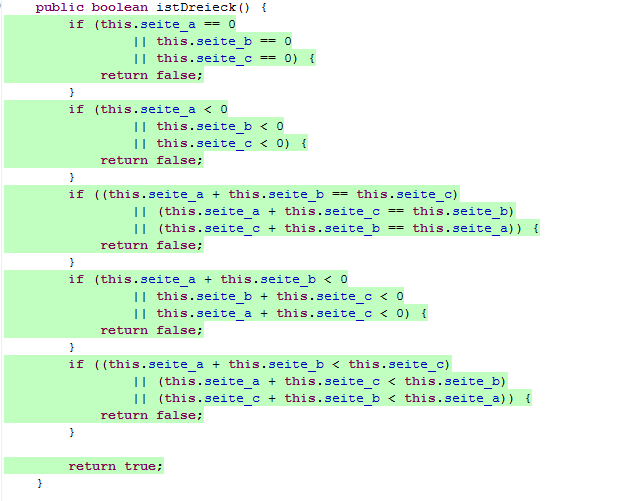
# Dreieck Tests – Github

## EclEmma Plugin installieren

* 1. Help => Eclipse Marketplace
  2. Nach EclEmma suchen
  3. Installieren
  4. Man kann nun bei einer Testklasse mit einem neuen Button die coverage checken bei der CUT (Class under Test) 
     1. Grün => Alle möglichkeiten gecovered
     2. Gelb => Ein paar Möglichkeiten gecovered
     3. Rot => Garkeine Möglichkeiten gecovered

## Projekt erstellen

Zuerst wird in Eclipse ein neues Projekt erstellt. In diesem Projekt werden zunächst 2 Source ordner angelegt:

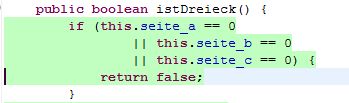
* Src => Ordner wo Dreieck.java liegt
* Test => Ordner wo die Testklassen liegen

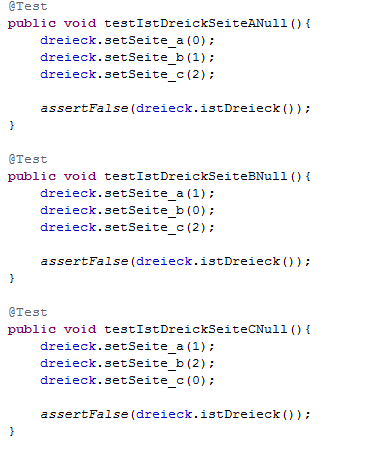
Dann wurden die nötigen Klassen erstellt / hinzugefügt sowie die Junit Library (Rechtsklick => Java build path => add library => JUnit)

## Git

1. Lokales repository anlegen mit *git init*
2. Remote repository hinzufügen mit *git remote add origin* [*https://github.com/mwoelfer-tgm/SEW\_15-16*](https://github.com/mwoelfer-tgm/SEW_15-16)
3. Synchronisieren mit *git pull origin master*
4. Files hinzufügen mit *git add IstDreieckTest.java*
5. Commiten mit *git commit -m „Add class for testing“*
6. Jedesmal wenn erneuerung hinzugefügt Schritte 4 und 5 wiederholen
7. Mit *git push* alles auf den externen Server „speichern“

## Testklasse hinzufügen: TestIstDreieck

1. Man schaut in der CUT nach was in der Methode istDreieck steht und stellt dann TestMethoden dafür auf
2. Beispiel: 
   * + - Es gibt 6 sogenannte „branches“:
         1. Seite A = 0
         2. Seite A!= 0
         3. Seite B = 0
         4. Seite B!= 0
         5. Seite C = 0
         6. Seite C!= 0



Mit der ersten Methode wurden gleich 3 Branches gededeckt:

* Seite A = 0
* Seite B!= 0
* Seite C!= 0

Mit der nächsten Methode wurden dann zwei weiter Branches gedeckt:

* Seite A!= 0
* Seite B = 0

Mit der letzten Methode wurde schließlich der letzte branch gedeckt:

* Seite C = 0

## istDreieck()

### testIstDreieckSeiteNull

Es wurde jede Seite mindestens einmal auf 0 und nicht 0 gesetzt, damit wurden die ersten Überprüfungen der Methode istDreieck gedeckt

### testIstDreickSeiteMinus

Diesmal wurde jede Seite mind. einmal größer 0 und kleiner 0 gesetzt.

### testIstDreieckSeiteAPlusBGleichC

Mit diesen Methoden wurde der nächste Block in der Dreieck Klasse überprüft. Nur mit der ersten Methode wurden schon 3 von 6 branches gedeckt:

* A + B = C
* A + C!= B
* C + B!= A

Mit der nächsten wurden folgende Branches gedeckt

* A + C = B
* A + B!=C

Mit der letzten Methode wurde der letzte branch gedeckt

* C + B = A

### testIstDreieckSeiteAPlusBMinus

Mit diesen Methoden musste man erzwingen dass

A + B < 0 usw. sein soll. Nun konnte man nicht einfach eines von beiden auf eine minuszahl setzen weil es schon bei der Bedingung davor „gecatched“ werden würde. Also musste man mit Overflow arbeiten, weil 1 + Integer.MAX\_VALUE = INTEGER.MIN\_VALUE, und diese Zahl ist minus

Das Prinzip zum decken aller branches ist das gleiche, man muss jede Bedingung einmal auf true und false haben.

### TestIstDreieckSeiteAPlusBKleinerC

Wie die methodennamen schon sagen, musste man hier lediglich erzwingen dass 2 Seiten zusammen kleiner eine andere sind.

### TestIstDreieckTrue

Ganz zum schluss kommt diese Methode noch weil wir bis jetzt die ganze zeit nur erzwungen haben das die Methode false zurück gibt, doch wir wollen auch diesmal beide branches decken deswegen einmal auf true