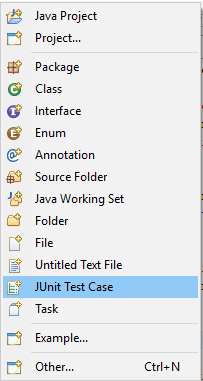
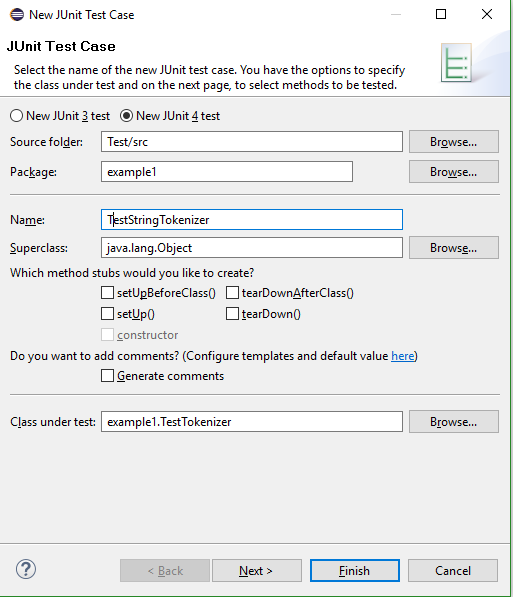
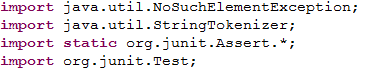
**Protokoll 01: Test StringTokenizer()**



Zuerst wird einen neue Klasse als   
JUnitTestCase erstellt.

File -> New -> JUnit Test Case

**Imports:**



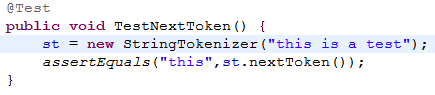
Import static org.junit.Assert.\*;  
bewirkt, dass man statt Klasse.Methode() nurnoch Methode() schreiben muss.

**Tests:**

**Um eine Methode mit JUnit zu testen wird @Test vor diese geschrieben.**

Token: Sind eine beliebige Anzahl an Zeichen bis zum nächsten Trennzeichen(Delimiter).  
nextToken(): gibt das nächsten Token aus.  
assertEquals(): prüft 2 Parameter auf ihre Gleichheit (Param1= Param2).

Der erste Test handelt sich um die Methode nextToken()



In diesem Test wird der StringTokenizer mit „this is a test“ befüllt und anschließend wird mit assertEquals verglichen, ob st.nextToken() auch „this“ zurückgibt. Falls dieser Fall eintritt ist der Test richtig.

Die nächsten Tests zeigen die Reaktion von nextToken() mit Delimitern:

Benutzte Delimiter:

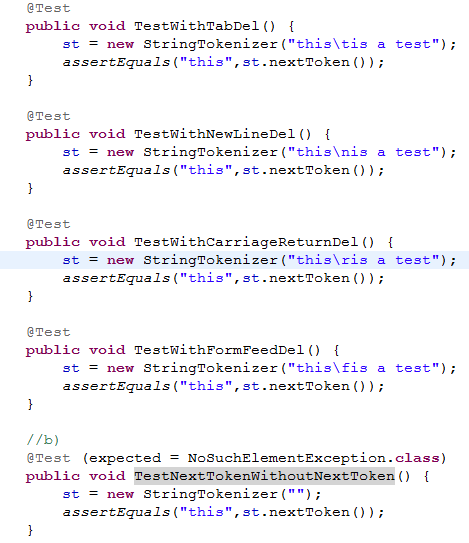
\t … Tabulator

\f … Nächste Seite

\r … Geht zum Beginn der Zeile

\n … Nächste Zeile

Dafür werden folgende Methoden erstellt:

hier wird \t anstatt des Leerzeichens also ein Tabulator eingefügt

hier wird \n anstatt des Leerzeichens also ein Zeilenumbruch eingefügt

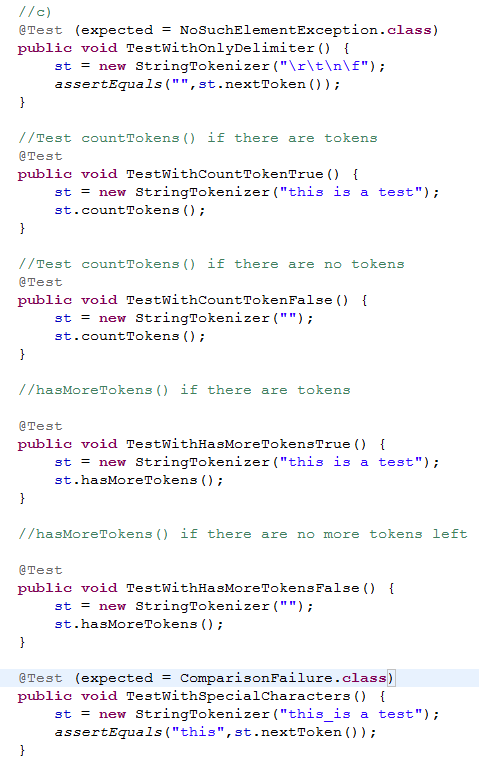
hier wird \r anstatt des Leerzeichens eingefügt, der Cursor geht also zum Anfang der Zeile

hier wird \f anstatt des Leerzeichens also ein Seitenumbruch eingefügt

hier wird getestet was passiert, falls es kein nextToken() gibt.

Dieser Test würde fehlschlagen, da ja kein nextToken vorhanden ist. Daher muss man JUnit manuell mitgeben, dass der Test in Falle der Exception eben kein Fehlschlag sondern ein Erfolg ist.

Da ein NoSuchElement Exception erwartet wird, wird dieser mit (expected = Exceptionname.class) als Erfolg des Tests gewertet.

 hier wird getestet, wie sich nextToken() verhält, wenn man nur Delimiter hat. Da bei diesem Test wieder eine Exception erwartet wird muss man diesen mit (expected) anschreiben.

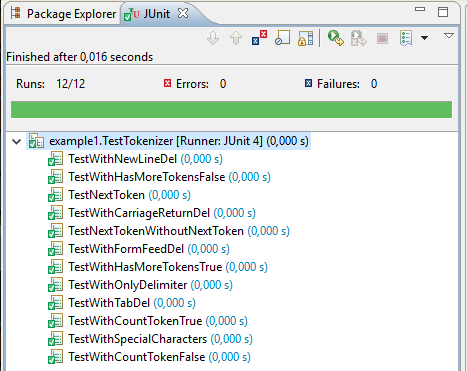
Hier wird die Methode countTokens() getestet.

Hier wird countTokens() ohne Tokens getestet. Hier muss man keine Exception erwarten, da die Rückgabe einfach 0 sein sollte.

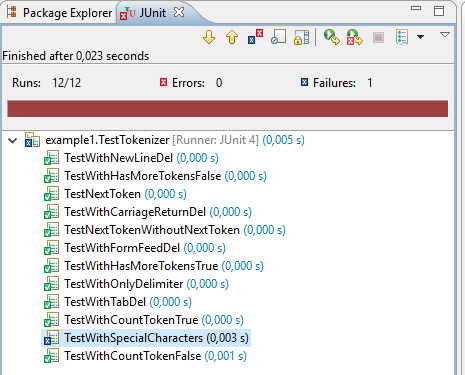
Hier wird die Methode hasMoreTokens() getestet

Hier wird die Methode hasMoreTokens() getestet falls es keine Tokens gibt. Man muss keine Exception erwarten, da die Methode einfach false zurück gibt.

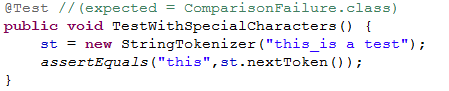
Hier wird getestet was passiert, wenn assertEquals einen Fehler beim Vergleichen findet.



Da jede Exception bei den Testfällen erwartet und als Erfolg gewertet wurde sieht das Endergebnis so aus:



Hier wurde bei dem TestWithSpecialCharacters-Test kein (expected= ….) daugeschrieben. Daher gibt eclipse den Test als fehlgeschlagen aus.



Dazugehöriger Quelltext Eclipse zeigt die aufgetretene Exception,

(expected auskommentiert) welche zum Felschlag geführt hat auf.