Unit-Tests sollen F.I.R.S.T. sein:

- Fast
- Isolated/Independent
- Repeatable
- Self-validating
- Timely (written)

Einschub Dreieck-Beispiel

Eine Methode `public String triangle(int a, int b, int c)`
gibt zurück ob ein Dreieck mit den Seitenlängen `a, b, c`

- gleichseitig -> “`equilateral`“
- gleichschenklig -> “`isosceles`“
- ein normales Dreieck -> “`scalene`“

ist

Welche Tests würden Sie schreiben um diese Methode zu testen?

(Glen Myers, The Art of Software Testing)

Wir brauchen:

1. Einen Test für ein gleichseitiges Dreieck
2. Einen Test für ein gleichschenkliges Dreieck
3. Einen Test für ein normales Dreieck
4. Mindestens 3 Testfälle für (2)! Alle Permutationen
5. Einen Test bei dem eine Seite 0 ist
6. Einen Test bei dem eine Seite negativ ist
7. Einen Test bei dem eine Seite die Summe der anderen beiden ist
8. Mindestens 3 Testfälle für (7)! Alle Permutationen
9. Einen Test bei dem eine Seite > die Summe der anderen beiden
10. Mindestens 3 Testfälle für (9)! Alle Permutationen
11. Einen Test bei dem alle Seiten 0 sind
12. (Einen Test mit nicht-ganzzahligen Eingaben, in Java unnötig)
13. (Einen Test mit der falschen Anzahl Parameter, in Java unnötig)
14. Hast du jeweils die erwartete Ausgabe spezifiziert?
15. Einen Test mit null

Noch mehr Randfälle

- **Falsehoods Programmers Believe About Names**
  https://shinesolutions.com/2018/01/08/falsehoods-programmers-believe-about-names-with-examples/
  - People’s names do not change.
  - People’s names do not contain numbers.
  - People’s names are assigned at birth.
  - People have names.

- **Falsehoods programmers believe about time**
  https://gist.github.com/timvisee/fcda9bbdff88d45cc9061606b4b923ca

Beispiel Mock

```
MyList listMock = Mockito.mock(MyList.class);
when(listMock.add(anyString())).thenReturn(false);

boolean added = listMock.add(randomAlphabetic(6));
assertThat(added, is(false));
```

Quelle Baeldung