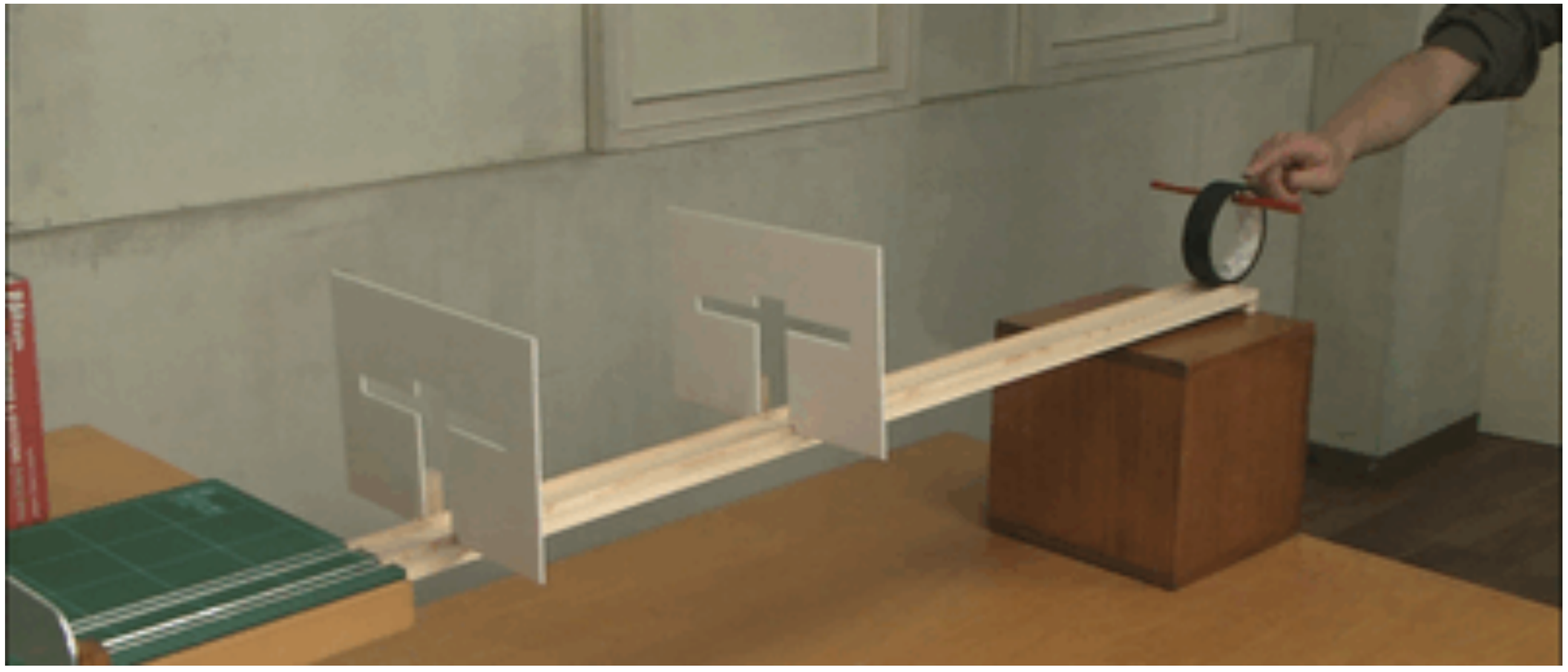
Testen

http://www.vogella.com/tutorials/Mockito/article.html

Warum testen?



Würde dies funktionieren, ohne es vorab zu testen?

Definition von Tests

- Ein Softwaretest prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten Anforderungen und misst ihre Qualität. Die gewonnenen Erkenntnisse werden zur Erkennung und Behebung von Softwarefehlern genutzt. Tests während der Softwareentwicklung dienen dazu, die Software möglichst fehlerfrei in Betrieb zu nehmen.

Arten von Tests

- Unit-Test: Test der einzelnen Methoden einer Klasse
- Integrationstest: Tests mehrere Klassen / Module
- Systemtest: Test des Gesamtsystems (meist GUI-Test)
- Akzeptanztest / Abnahmetest: Test durch den Kunden, ob Produkt verwendbar
- Regressionstest: Nachweis, dass eine Änderung des zu testenden Systems früher durchgeführte Tests erneut besteht
- Feldtest: Test während des Einsatzes
- Lasttest
- Stresstest
- Smoke-Tests: (Zufallstest) nicht systematisch; nur Funktion wird getestet

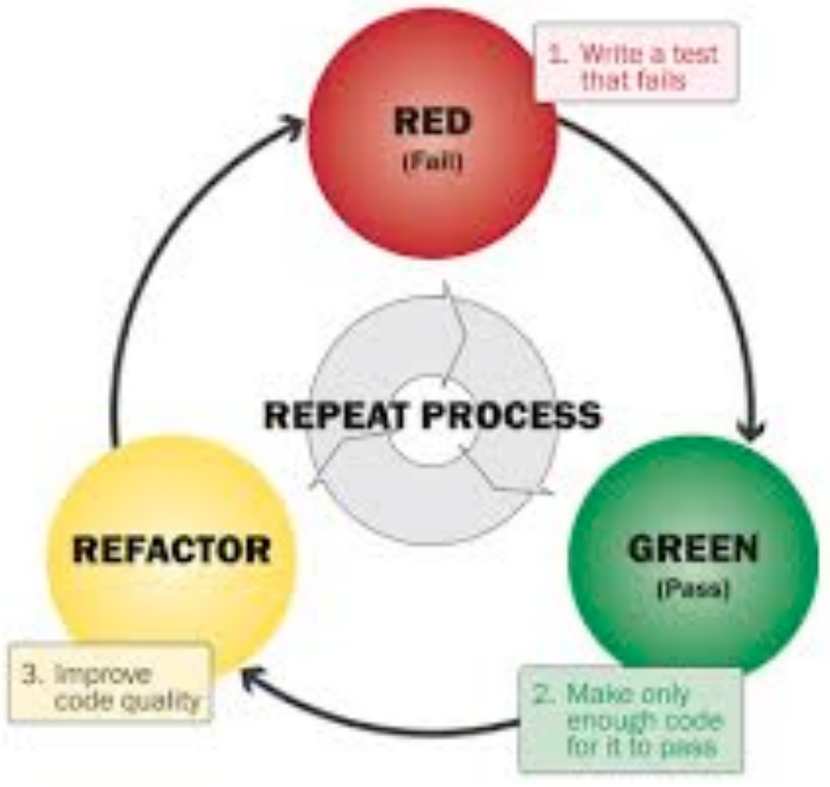
Validierung: Mache ich die richtigen Dinge?
WAS

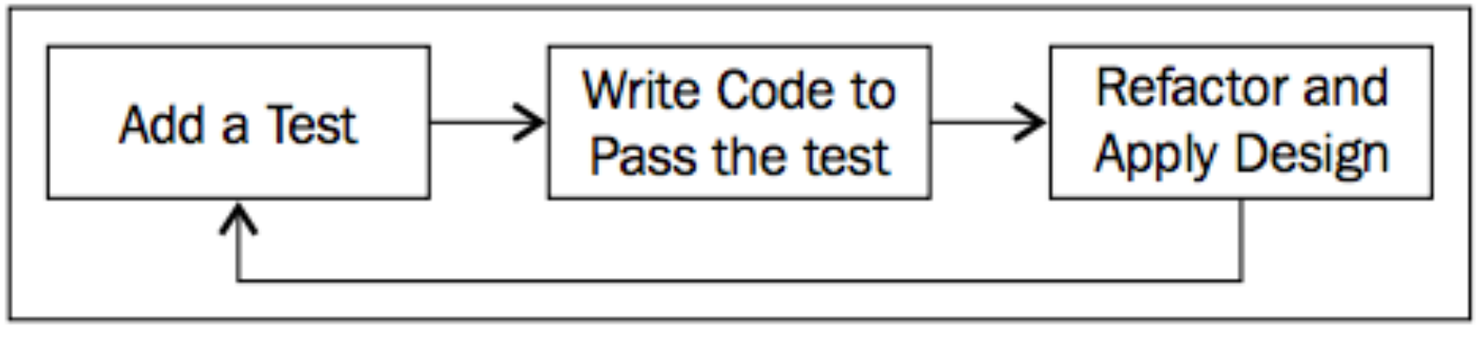
Verifizierung: Mache ich die Dinge richtig
WIE

Definitionen

- SUT … system under test
- CUT … class under test

TDD - Test Driven Development





Test-Driven Development with Mockito, packt 2013

JUnit.org

junit.org/index.html

Search

Overview Crowdfunding Project Documentation

Fork me on GitHub

JUnit

JUnit / About

Version: 4.12 | Last Published: 2015-09-24

JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.

```
@Test
public void lookupEmailAddresses() {
    assertThat(new CartoonCharacterEmailLookupService().getResults("looney"), allOf(
        not(empty()),
        containsInAnyOrder(
            allOf(instanceOf(Map.class), hasEntry("id", "56"), hasEntry("email", "roadrunner@fast.org")),
            allOf(instanceOf(Map.class), hasEntry("id", "76"), hasEntry("email", "wiley@acme.com"))
        )
    ));
}
```

Hamcrest matchers

Make your assertions more expressive and get better failure reports in return.

Let's take a tour »

Welcome

Usage and Idioms

Third-party extensions

Hamcrest

- Hamcrest is a library of matchers, which can be combined in to create flexible expressions of intent in tests. They've also been used for other purposes

- http://hamcrest.org/
- https://github.com/hamcrest/JavaHamcrest
- http://www.leveluplunch.com/java/examples/hamcrest-collection-matchers-junit-testing/
- http://grepcode.com/file/repo1.maven.org/maven2/org.hamcrest/hamcrest-library/1.3/org/hamcrest/Matchers.java

Dependencies

```xml
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>2.6.1</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>2.1.0</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.derby</groupId>
    <artifactId>derbyclient</artifactId>
    <version>10.12.1.1</version>
    <scope>compile</scope>
</dependency>
```

Testing private methods

There are basically 4 options:

- Don't test private methods.
- Give the methods package access.
- Use a nested test class.
- Use reflection.

Eine Meinung …

- I have used reflection to do this in the past, and in my opinion it was a big mistake.

- Strictly speaking, you should not be writing unit tests that directly test private methods. What you should be testing is the **public contract that the class has with other objects**; you should never directly test an object's internals. If another developer wants to make a small internal change to the class, which doesn't affect the classes public contract, he/she then has to modify your reflection based test to ensure that it works. If you do this repeatedly throughout a project unit tests and then stop being a useful measurement of code health, and start to become a hindrance to development, and an annoyance to the development team.

- What I recommend doing instead is using a code coverage tool such as Cobertura, to ensure that the unit tests you write **provide decent coverage of the code in private methods**. That way, you indirectly test what the private methods are doing, and maintain a higher level of agility.

http://stackoverflow.com/questions/34571/how-to-test-a-class-that-has-private-methods-fields-or-inner-classes

mit Reflection …

Klasse: at.htl.business.BookingController

```
private int add(int a, int b) {
    return a + b;
}
```

```
private int at.htl.business.BookingController.add(int,int)
```

Test:

```
@Test
public void t030testPrivateMethod() throws Exception {
    Class<?> cut = Class.forName("at.htl.business.BookingController");
    System.out.println("**********************************************************");
    Method[] methods = cut.getDeclaredMethods();
    for (Method m : methods) {
        System.out.println(m);
    }
    System.out.println("**********************************************************");

    final Class<?>[] paramTypes = new Class<?>[] {int.class, int.class};
    Method privateMethod = cut.getDeclaredMethod("add", paramTypes);
    privateMethod.setAccessible(true);
    int result = (int) privateMethod.invoke(bookingController,3,8);
    assertThat(11, is(result));
}
```

Underlying object

Links

- http://www.leveluplunch.com/java/examples/hamcrest-collection-matchers-junit-testing/
- https://objectpartners.com/2013/09/18/the-benefits-of-using-assertthat-over-other-assert-methods-in-unit-tests/

Noch Fragen?