

Software Horror Stories

Heute: Slammer

Buffer Overflow

```
int[] a = new int[10];  
for (int i = 0; i < 100; i++) {  
    a[i] = i;  
}
```

- Was passiert in Java?
- Was passiert bei einem äquivalenten C Programm?

Buffer Overflow

```
int main(void) {
    char buff[15];
    int pass = 0;
    printf("\n Enter the password : \n");
    gets(buff);
    if(strcmp(buff, "foobar")) {
        printf ("\n Wrong Password \n");
    }
    else {
        printf ("\n Correct Password \n");
        pass = 1;
    }
    if(pass) {
        printf ("\n Root privileges given to the user \n");
    }
    return 0;
}
```

DEMO

Buffer Overflow

- Das ist noch nicht das Schlimmste!
- Mit Buffer Overflows kann man auch beliebigen Code ausführen
- Wer mehr darüber lernen will: Netzwerksicherheit (Master)

Beispiel: MS SQL Server

- Verzeichnisdienst zur Suche nach Datenbanken im Netzwerk
- Empfängt ein Netzwerk-Paket mit einem maximal 16 Zeichen langen Namen über das Netzwerk
- Der Name wird mit zwei konstanten Strings auf dem Stack konkateniert
- Der Buffer auf dem Stack hat 128 Byte
- Problem: Die Länge des Packets wurde nicht gecheckt

Wurm

- Ein Wurm ist ein Programm, das sich selbst repliziert ohne dass der Nutzer involviert ist
- Normalerweise verwendet ein Wurm eine Sicherheitslücke um auf einem Rechner ausgeführt zu werden
- Dann versucht der Wurm andere Computer im Netzwerk zu infizieren

Bekannte Würmer

- Morris (1988): Erster Internet Wurm, multiple Angriffsvektoren (sendmail, finger, rsh/rexec, weak passwords)
- Code Red (2001): Angriffsvektor Microsoft IIS, DoS Attacke auf verschiedene Websites, Defacement der eigenen Seite, Installation einer Backdoor, Infektion von 250.000 Systemen in ca. 9 Stunden

25. Januar 2003

Ein 376 Bytes großes
Netzwerkpaket macht
sich auf die Reise

Slammer

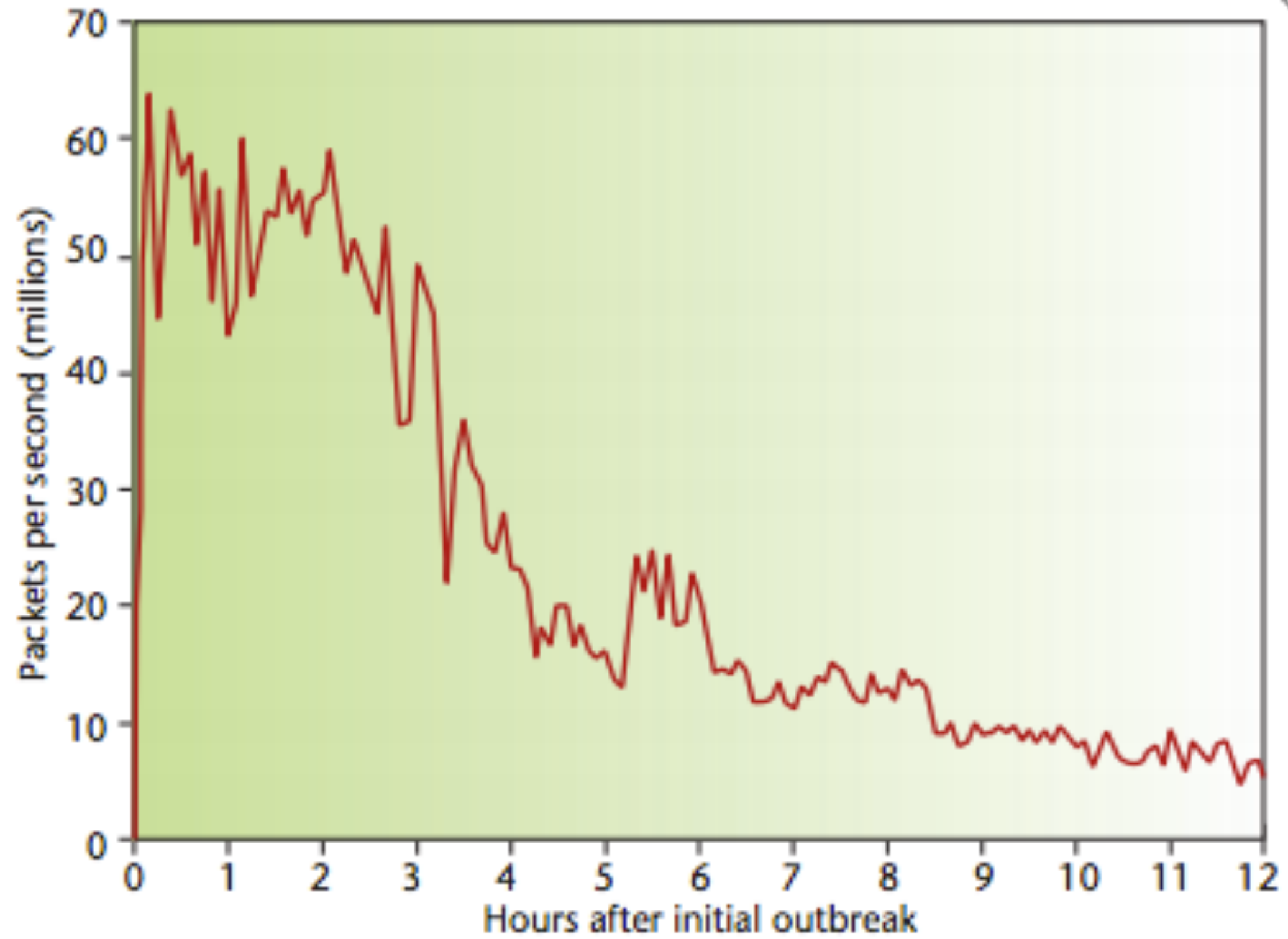
- Angriffsvektor: MS SQL Server
- Der Wurmcode wird an den Verzeichnisdienst geschickt
- Es gibt einen Buffer Overflow und der Wurmcode wird ausgeführt
- Der Wurm geht in eine Endlosschleife:
 - Erzeuge eine 32 Bit Zufallszahl
 - Interpretiere die Zahl als IP Adresse
 - Versende ein Paket mit dem Wurm an diese Adresse



25.01.2003 0:55 Uhr



25.01.2003 1:00 Uhr



Kurz nach der ersten
Infektion sind Teile des
Internets down

Folgen

- Ein großer Anteil der Bandbreite wurde von Slammer konsumiert (ironisch: Das kleine Paket wurde bevorzugt behandelt)
- Störungen der Notrufnummer
- Störungen von ATMs
- Störungen im Flugverkehr
- Störung der Sicherheitssysteme im Kernreaktor Davis-Besse, Ohio
- Gesamtschaden (Schätzung): > 1.000.000.000 \$

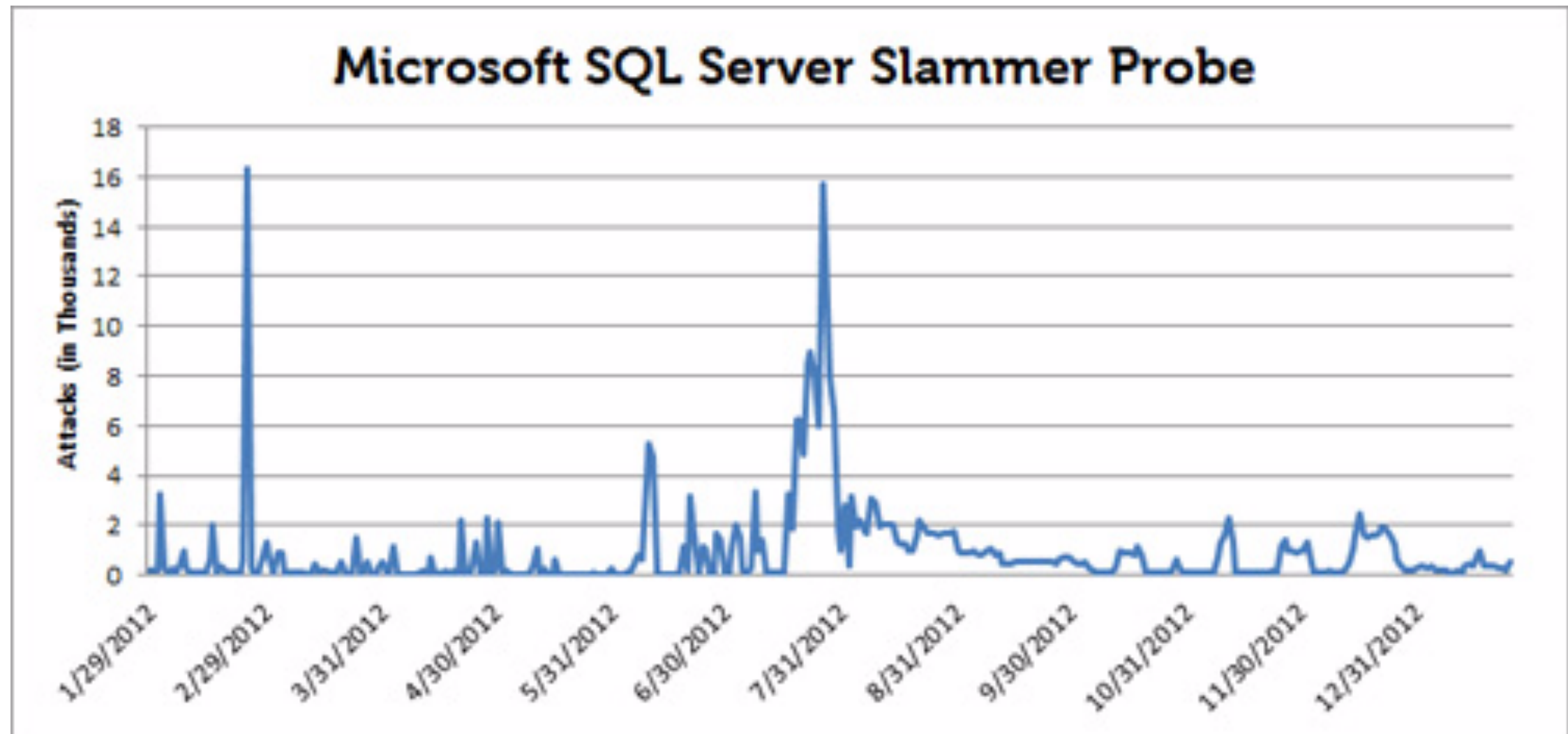
Effektivität

- Viele anfällige Systeme
- Klein (< 400 Bytes)
- UDP
- Extrem aggressiv
 - Infektion von 90% aller verletzlichen Systeme in 10 Minuten
 - Exponentielles Wachstum: Verdopplung alle 8.5 Sekunden
- Viele ungepatchte Systeme
(der Bug war ~ 6 Monate vorher behoben worden!)

Lessons learned

- Unbedingt auf Buffer Overflows achten, wenn man C programmiert
- Niemals gets verwenden, das ist inhärent kaputt!
- Software auf dem aktuellen Stand halten und Sicherheitspatches einspielen
- Nutzer patchen ihre Systeme nicht!

Slammer 2012



Über 10 Jahre nach dem Ausbruch ist Slammer immer noch Teil des Hintergrundrauschens im Netz

Feedback (EvaSys)

Auszüge Freitext

- Der Kurs ist nur für die machbar die schon programmieren können.

Teilnahmevoraussetzungen

Erfolgreicher Abschluss des Moduls:

- „Grundlagen der Softwareentwicklung und Programmierung“ (Informatik I)

Auszüge Freitext

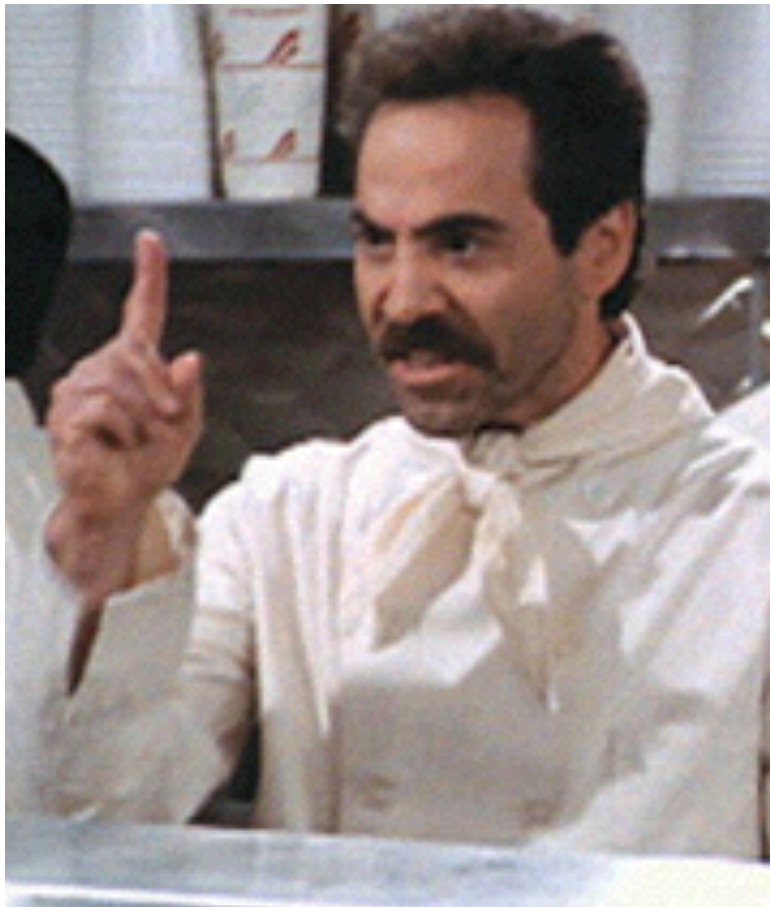
- Weder Vorlesung noch Übung bespricht sinnvoll die Themen die relevant für die Projekte sind.
- Ein Zusammenhang zwischen der Vorlesung, den Übungen den Aufgaben und der Klausur würde das Verständnis erleichtern.
- Der Besuch der Vorlesung bringt einen für die Bearbeitung der Projekte nicht wirklich weiter.

Koordination?

- Die Übungen waren in der Regel nicht relevant für das aktuelle Projekt
- Aber sie sind alle relevant für das Abschlussprojekt!
- Das gleiche gilt (mit Ausnahme von Projekt 1) für die Projekte
- Alle bisherigen Projekte hatten das Ziel Sie für das Abschlussprojekt fit zu machen

Auszüge Freitext

- da die Übungen nichts mit den Projekten zu tun haben, ist diese Veranstaltung sehr aufwändig und die Klausur wird viel zu schwer. [...]



No Klausur for you!

Auszüge Freitext

- Was auch problematisch ist, dass die Themen in den Übungen so neu sind, dass selbst die Tutoren sich nicht mit den Themen so gut auskennen teilweise.
 - Das ist ein absolut valider Punkt, es kann aber nicht so weitergehen wie bisher!
 - Es ist auch offensichtlich, warum ich nicht alle Übungen selber halten kann.
 - Das Problem sollte sich aber mit diesem Jahr erledigen.

Auszüge Freitext

- die Software Horror Stories sind sehr interessant
- Zu viel inhaltlicher Muell am Anfang.

Ich halte die Stories für wichtig. Sie schärfen den Blick für Fehler.

Auszüge Freitext

- Die Vorlesung darf insgesamt gerne etwas strenger sein. Das mögliche Lernpotential ist durchaus sehr hoch, jedoch habe ich das Gefühl, dass viele mit minimalen Anstrengungen diesen Kurs problemlos bestehen werden.

Auszüge Freitext

- Sehr interessant und lehrreich=)
- Mit Hilfe von sehr interessanten Beispielen aus dem Alltag schafft er es, den Studenten das Modul näher zu bringen
- Der Stil der Vorlesung ist sehr unterhaltsam und sorgt dadurch dafür, dass die Motivation und Aufmerksamkeit hoch bleibt beim Zuhören
- Die Themen der Übung ergänzen die Vorlesung sehr gut.

Das komplette
Ergebnis ist in Ilias

Projekt 5

Projekt 5

- Etwa die Hälfte von Ihnen muss das Projekt nachreichen
- Bei den bestandenen Projekten gibt es eine grosse Anzahl an Abgaben, die nur technisch korrekt sind
- D.h. sie erfüllen die Regeln, haben aber kein gutes OO Design
- Leider ist der Lerneffekt dann eher überschaubar!



Object Calisthenics done wrong

Quelle: <https://www.youtube.com/watch?v=F1spwQa-NUI>

Der Weg ist das Ziel, das
Ergebnis ist im Prinzip
egal!

Aber es gab viele tolle
Diskussionen in Ilias :-)

Und auch einige
ordentliche Abgaben

Projekt 6

Abgabe

- Es gab offenbar eine unzureichende Kommunikation meinerseits
- Die Tatsache, dass Sie nachreichen können, heisst nicht, dass Sie zum Stichtag nichts abgeben dürfen!
- Es soll zum Stichtag eine Abgabe vorhanden sein, die substantiell ist

Gründe

- **Ich möchte nicht, dass Sie die Projekte wie eine Bugwelle vor sich herschieben und am Ende des Semesters zu sehr unter Druck geraten**
- Die Abgaben geben mir einen Eindruck, wieviele von Ihnen noch mit an Bord sind
- Es ist sehr nervig die Abgaben im System manuell nachzupflegen

Wenn Sie morgen abgeben und
Sie wissen, dass Ihre Abgabe
nicht funktioniert:

Schreiben Sie das in das
Textfeld, damit der Korrektor
Bescheid weiss!

Abschlussprojekt

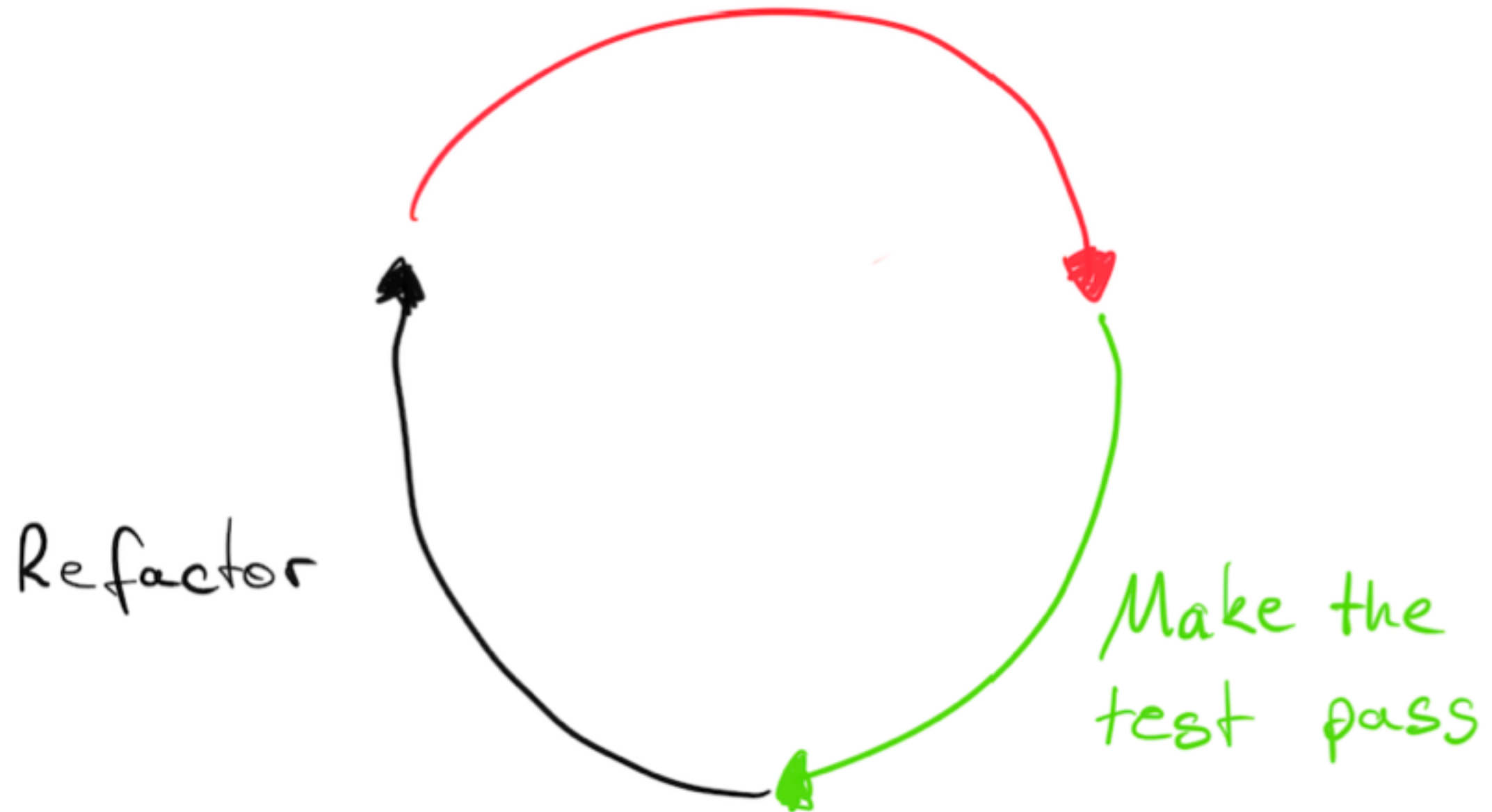
Team-Bildung

- Über Github Classroom, wie in dieser Woche in den Übungen
- Sie können neue Team bilden!
- Gruppengröße: 4 ± 1
- Bitte treten Sie nur dann in ein Team ein, wenn Sie das mit den anderen Mitgliedern abgesprochen haben
- Sie können die Teambildung im Orgaforum in Ilias absprechen, wenn Sie noch kein Team haben
- Mein Tipp: Wenn Sie Teammitglieder im Forum gefunden haben, treffen Sie sich persönlich

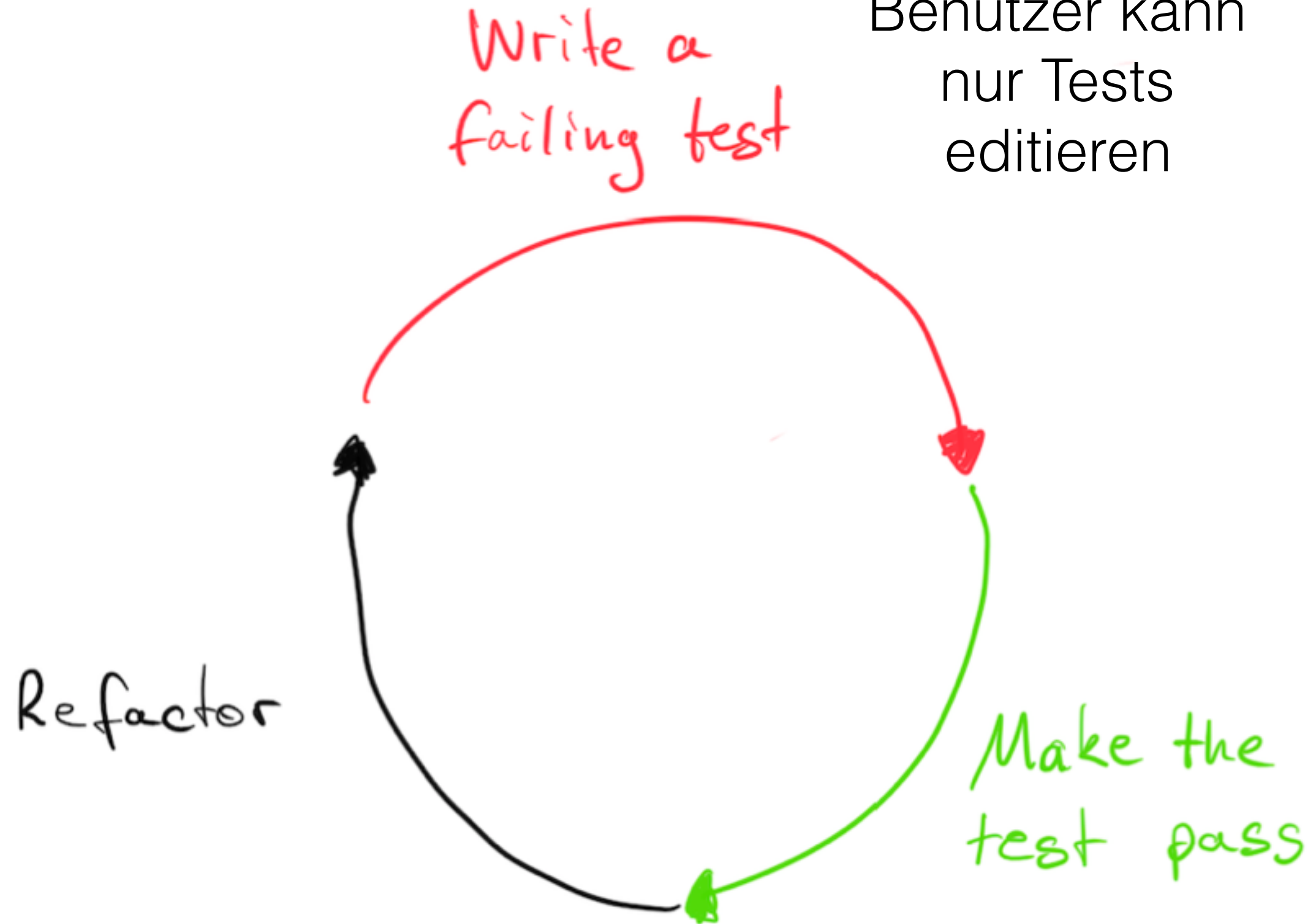
Das Abschlussprojekt

- TDDT - Test Driven Development Trainer
- Ein Programm mit dem Anfänger TDD lernen können
- Benutzer werden von dem Programm in dem TDD Zyklus gehalten
- Das folgende Konzept ist nicht vollständig, es dient nur der Vermittlung der Idee!

Write a
failing test



Benutzer kann
nur Tests
editieren

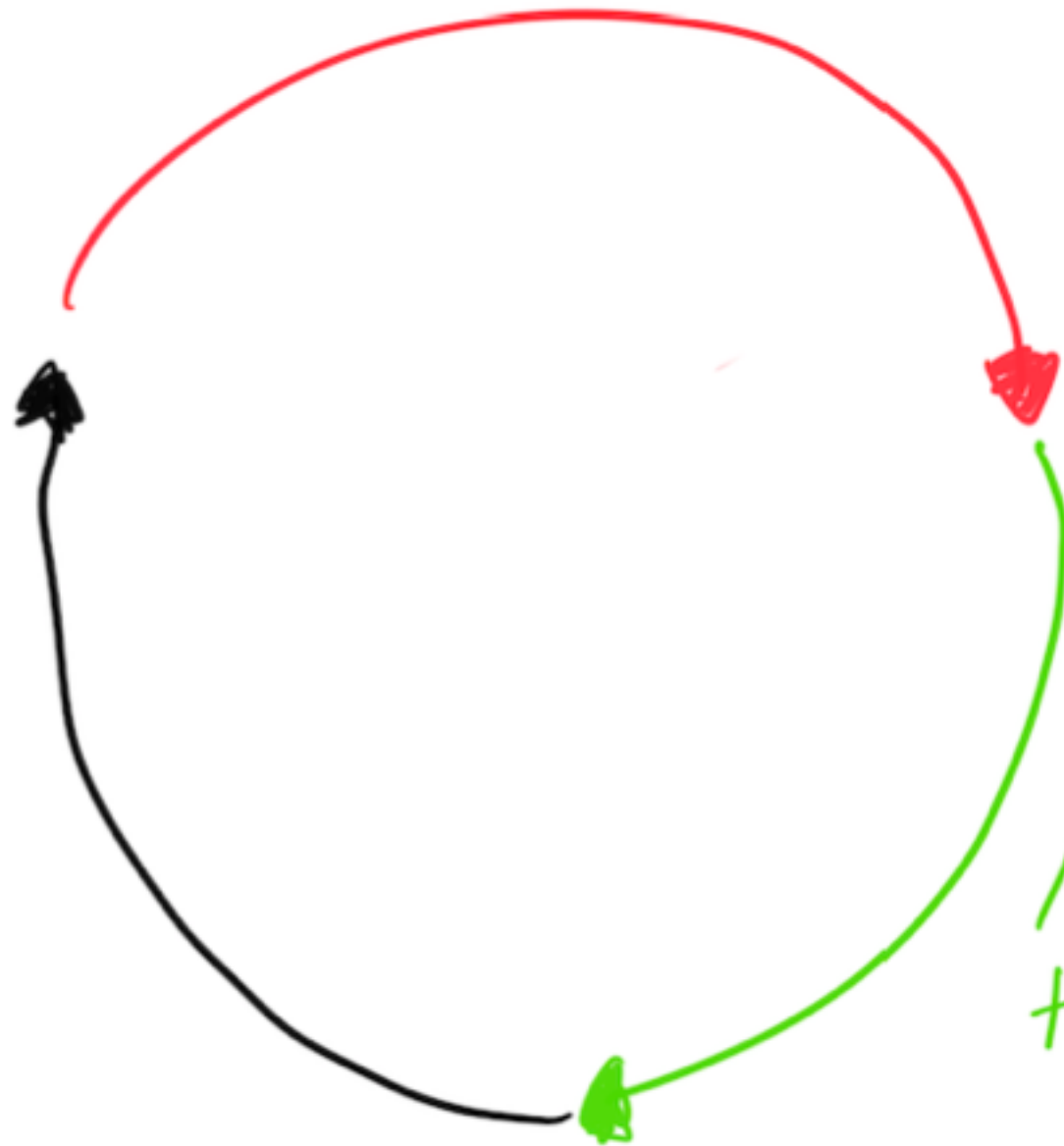


Write a
failing test

Benutzer kann
nur
weitermachen,
wenn genau
ein Test failed

Refactor

Make the
test pass



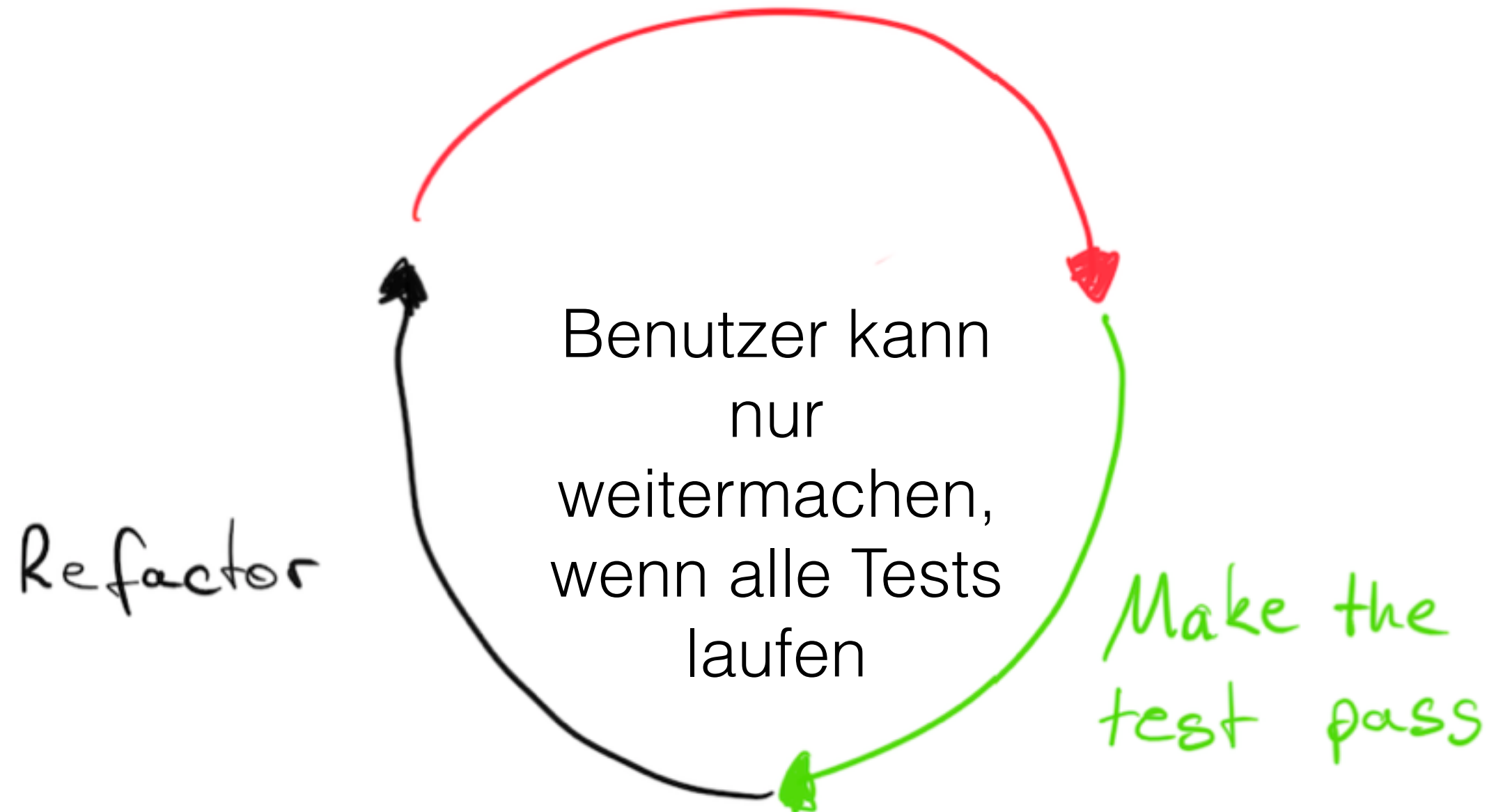
Write a
failing test

Refactor

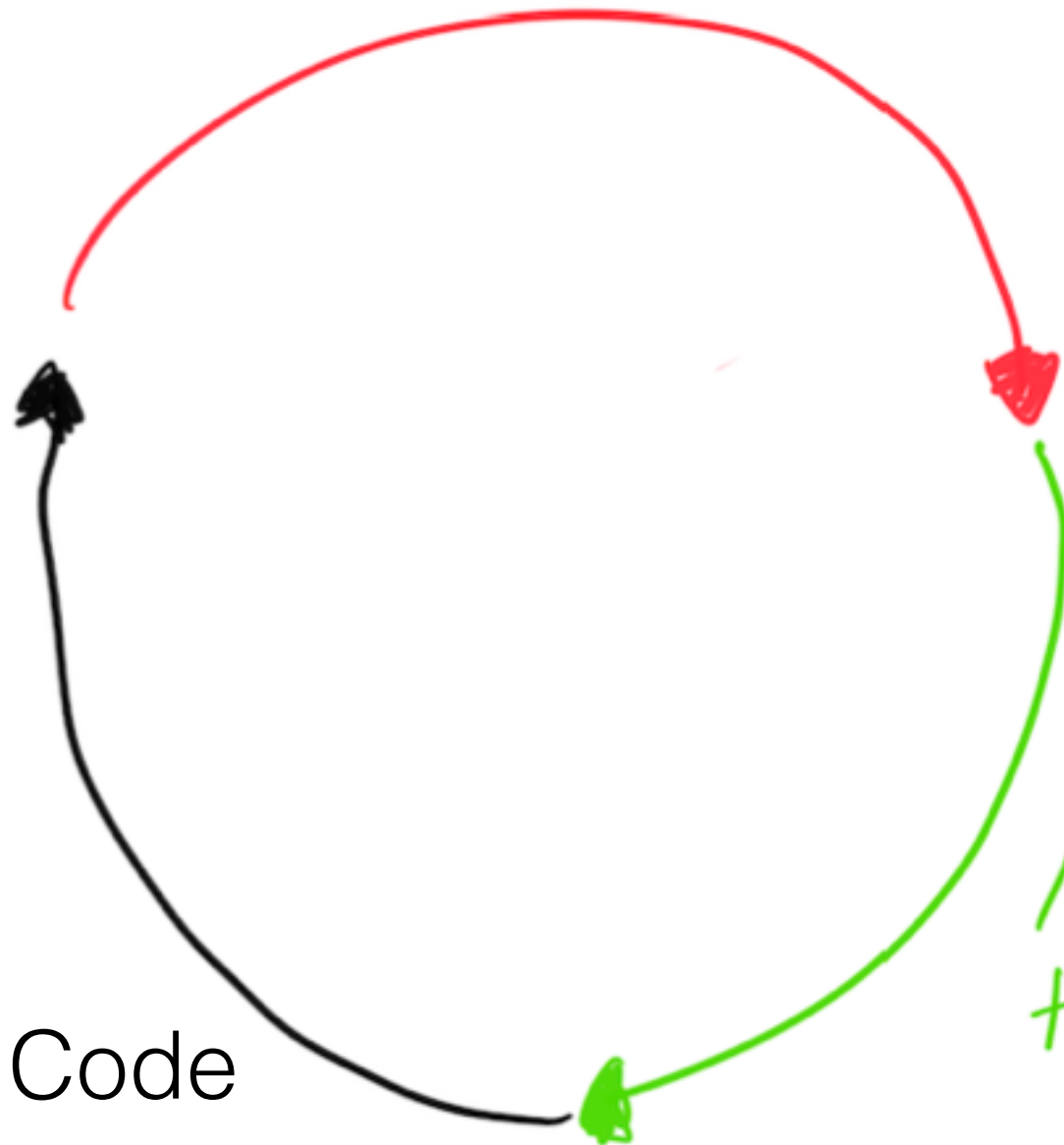
Make the
test pass

Benutzer kann
nur Code
editieren

Write a
failing test



Write a
failing test



Make the
test pass

Benutzer kann nur
Tests verbessern
Refactor

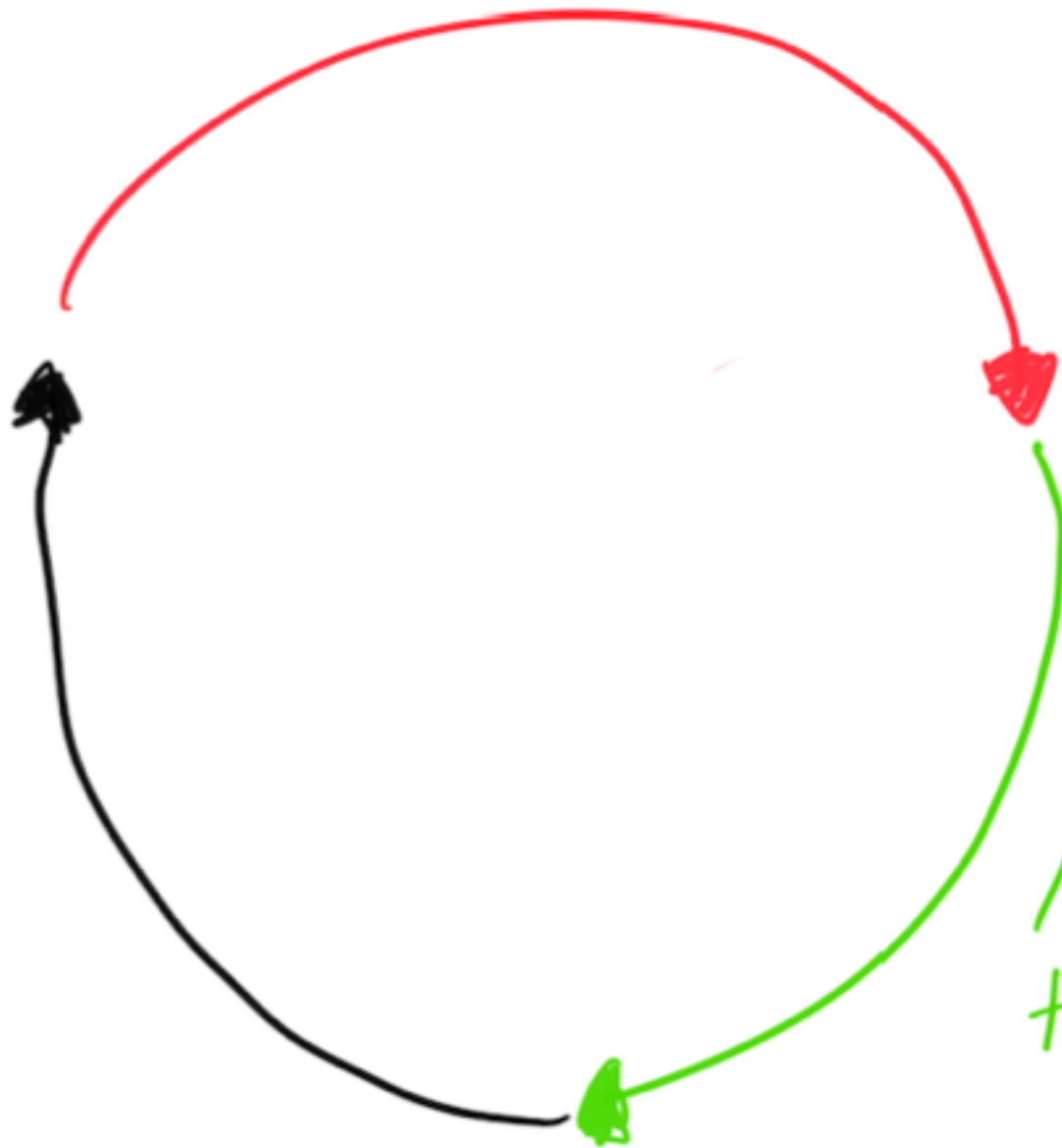
Benutzer kann nur Code
verbessern

Benutzer kann
nur
weitermachen,
wenn alle Tests
laufen

Refactor

Write a
failing test

Make the
test pass

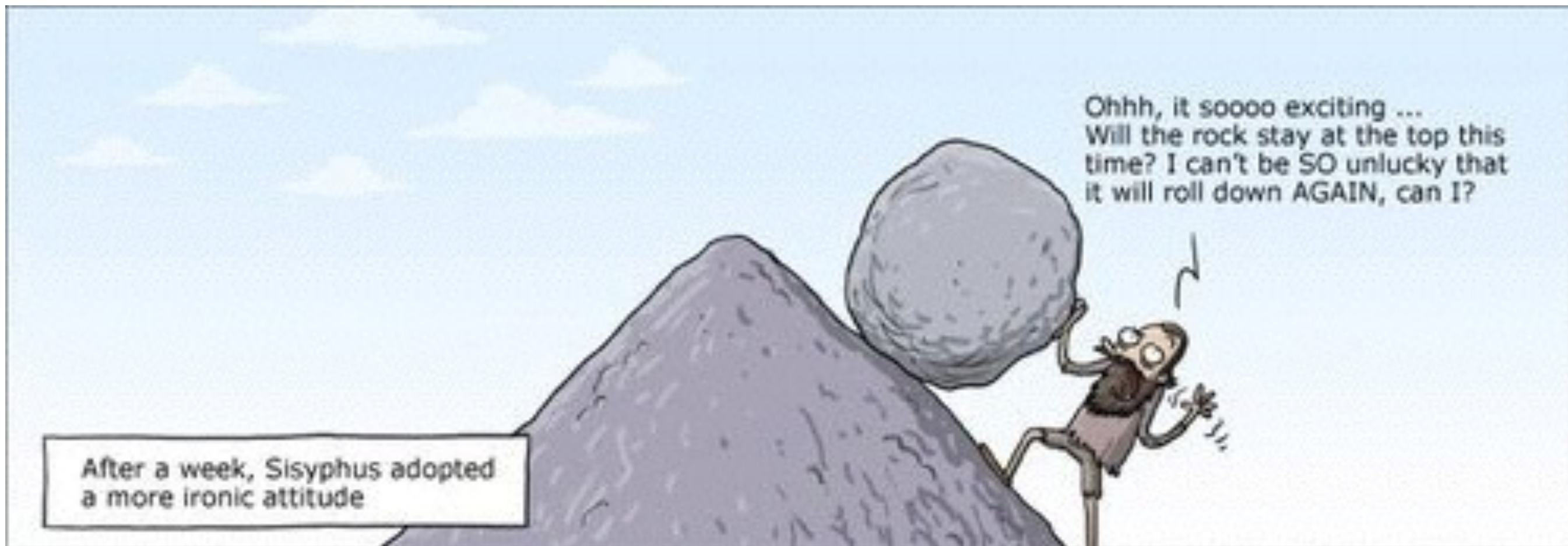


Compiler Bibliothek

- Sie brauchen keine Schnittstelle zum Compiler und Test-Runner zu implementieren, das habe ich für Sie getan
- Sie sollen den Prozess in einer GUI implementieren

Eat your own dog food

- Sie müssen kein Pair-Programming machen
- Sie müssen keine Tests schreiben
- Ich empfehle es aber nachdrücklich!
- Auf den ersten Blick scheint es mehr Arbeit zu sein, es wird Ihnen aber Zeit einsparen!



Entwicklungsprozess

- Entwicklung findet in git über Github statt
- Ihr Projekt wird mit Gradle gebaut
- Sie verwenden Travis für Continuous Integration
- Sie treffen sich mit Ihrer Gruppe mindestens einmal pro Woche und diskutieren Ihren Fortschritt
- Sie protokollieren diese Treffen
(jede Woche ein anderes Teammitglied)

Ein Wort zu git



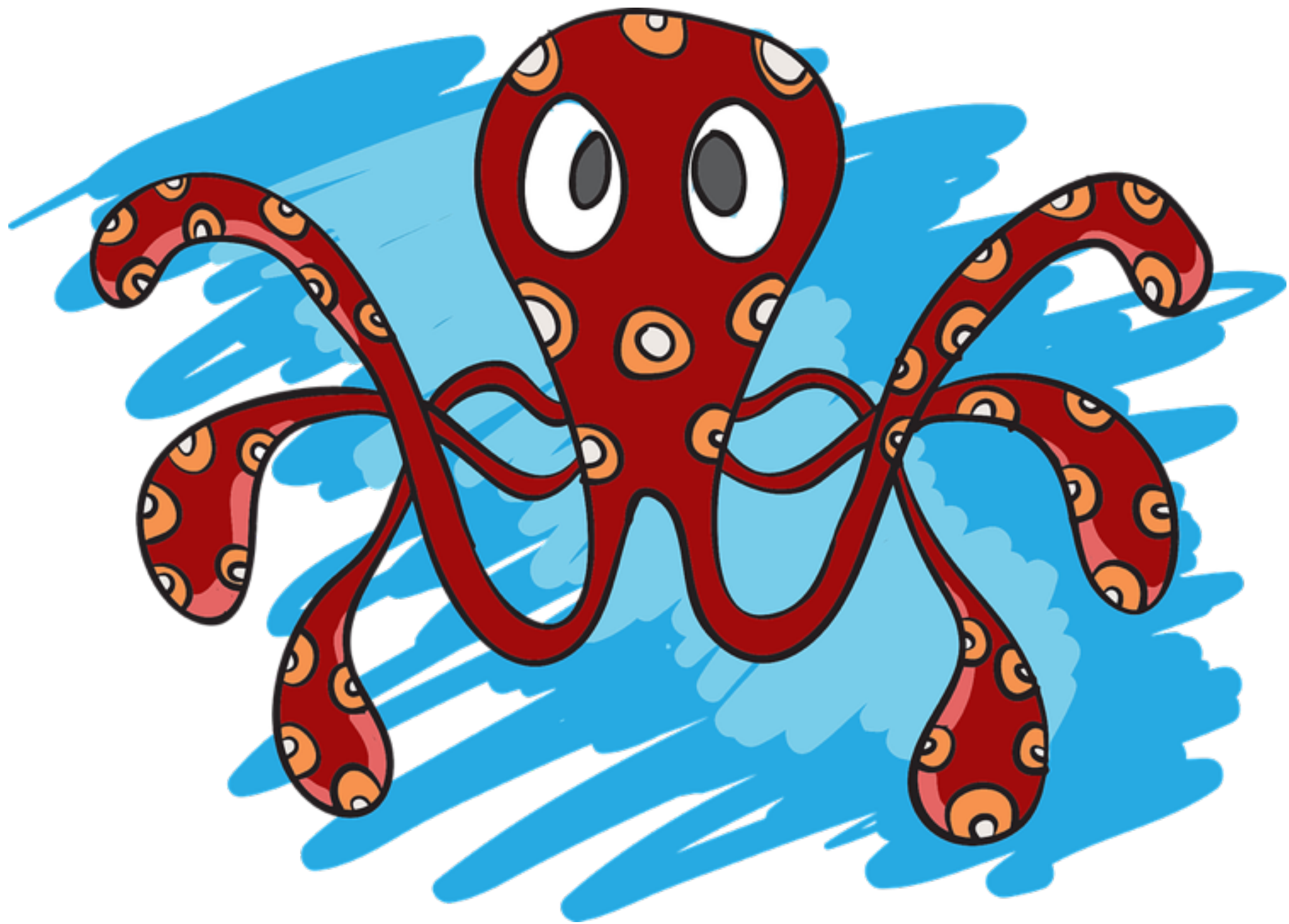
"The force you must not use!"

Master Yoda on git force push

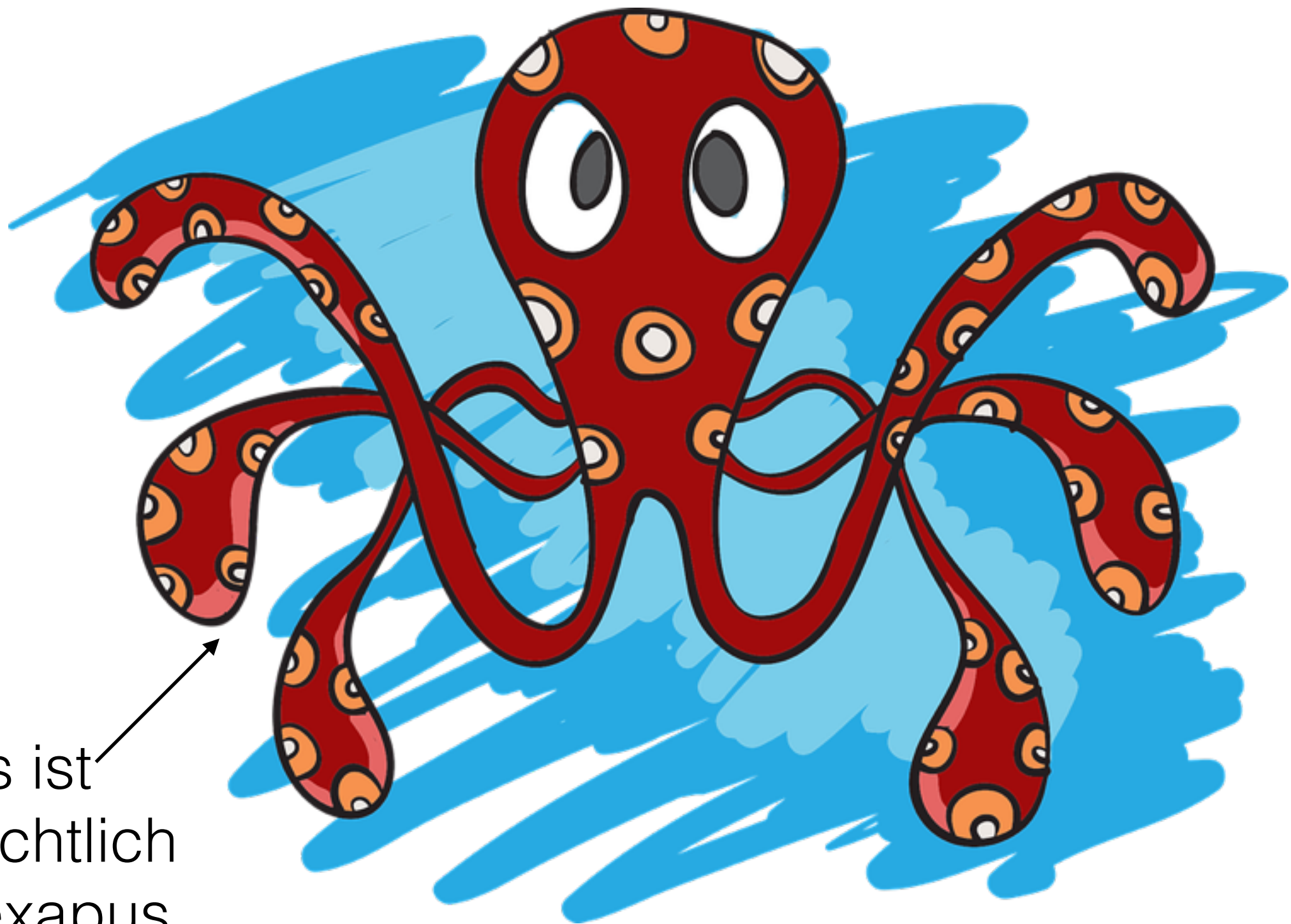
Zwei Worte
~~Ein Wort~~ zu git

Merge

- Jens' beliebtester Fehler:
`git merge feature/foo master`
- Was will Jens mit dem Befehl erreichen und was macht der Befehl wirklich?
- Jens wollte den Branch feature/foo in den master Branch mergen
- Tatsächlich merged der Befehl aber gleichzeitig master als auch feature/foo in den aktuellen Branch



Octopus Merge



Das ist
offensichtlich
ein Hexapus

Octopus Merge

Heute

- Zusammenhang: Unit Tests und Dependency Inversion
- Building
- Continuous Integration
- Das Abschlussprojekt

Unit Tests

Anatomie eines Tests: AAA

@Test

```
public void toRoman_given5_shouldReturnV () {  
    RomanConverter converter = new RomanConverter();  
    String result = converter.toRoman(5);  
    assertEquals("V", result);  
}
```

Arrange

@Test

```
public void toRoman_given5_shouldReturnV () {  
    RomanConverter converter = new RomanConverter();  
    String result = converter.toRoman(5);  
    assertEquals("V", result);  
}
```


Act

@Test

```
public void toRoman_given5_shouldReturnV () {  
    RomanConverter converter = new RomanConverter();  
    String result = converter.toRoman(5);  
    assertEquals("V", result);  
}
```

Assert

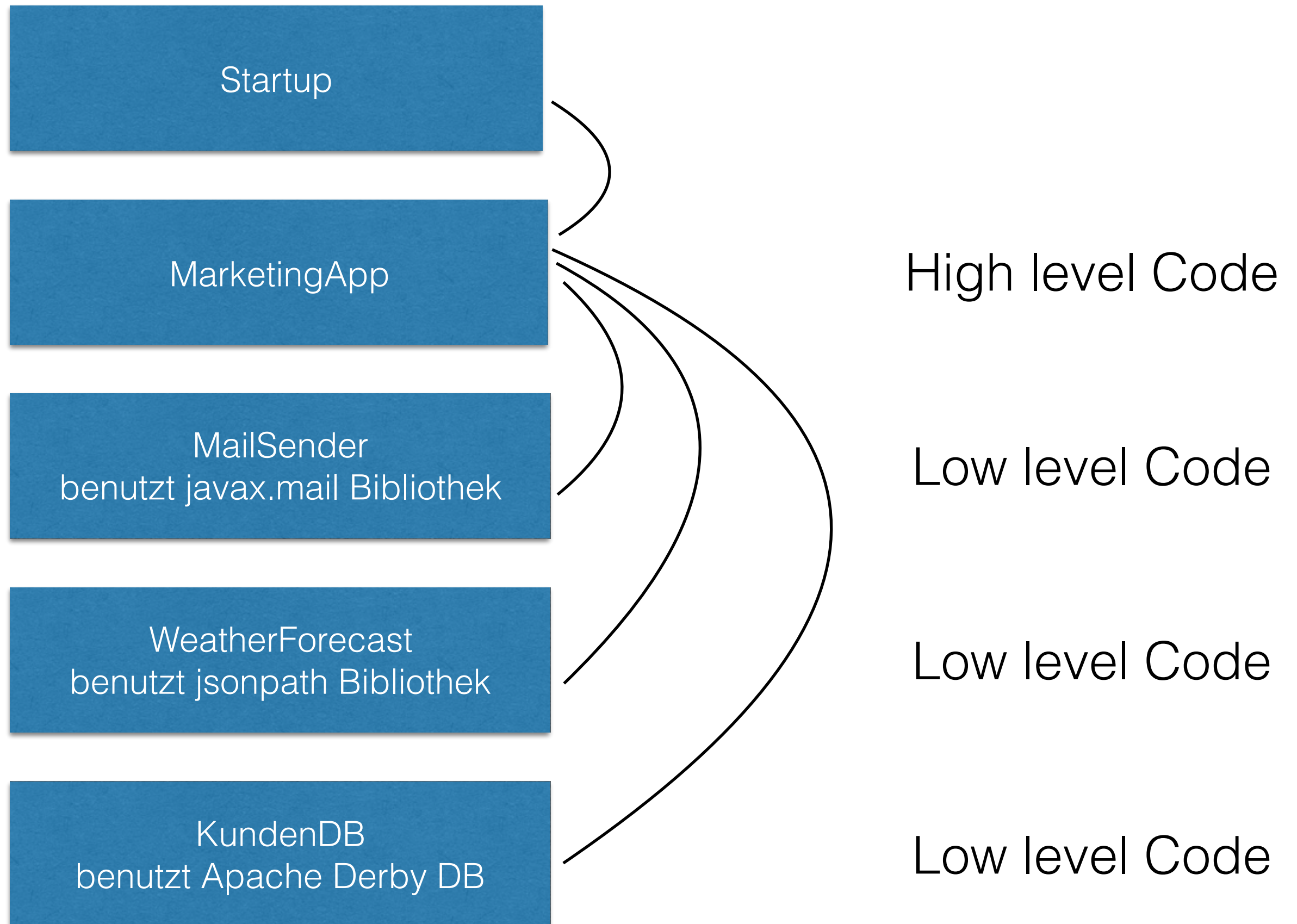
@Test

```
public void toRoman_given5_shouldReturnV () {  
    RomanConverter converter = new RomanConverter();  
    String result = converter.toRoman(5);  
    assertEquals("V", result);  
}
```

Beispiel:
Bendispostos
Bällchen
Bude

Beispiel

- Für unser Eiscafe wollen wir Marketing betreiben
- Wir haben schon eine Weile Mailadressen unserer Kunden gesammelt
- Wir wollen immer, wenn es in Düsseldorf über 30 Grad warm wird, eine Mail an unsere Kunden schicken und sie daran erinnern, dass es tolles Eis gibt



Was haben Unit-Tests mit
Dependency Inversion/
Injection zu tun?

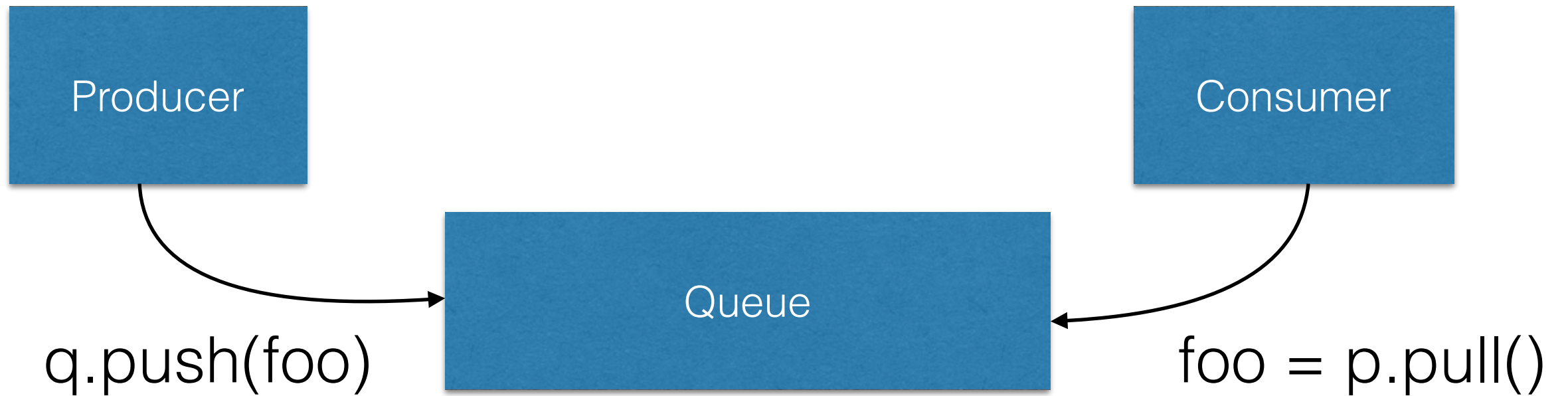
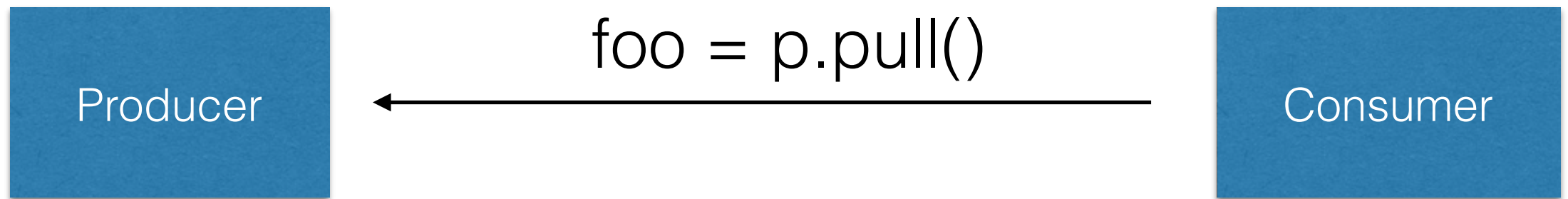
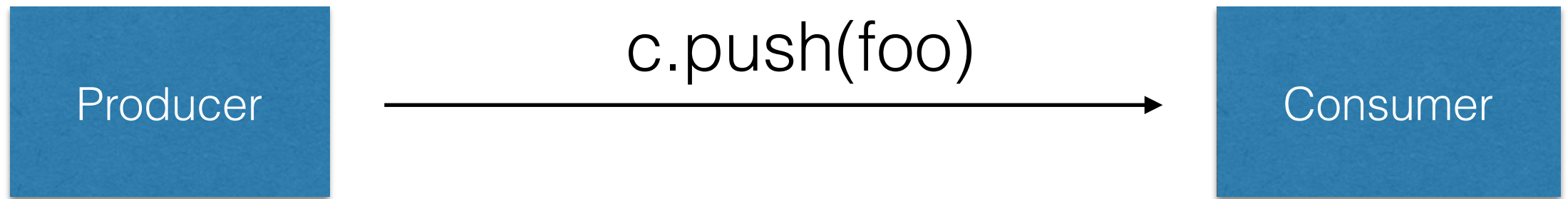
DEMO

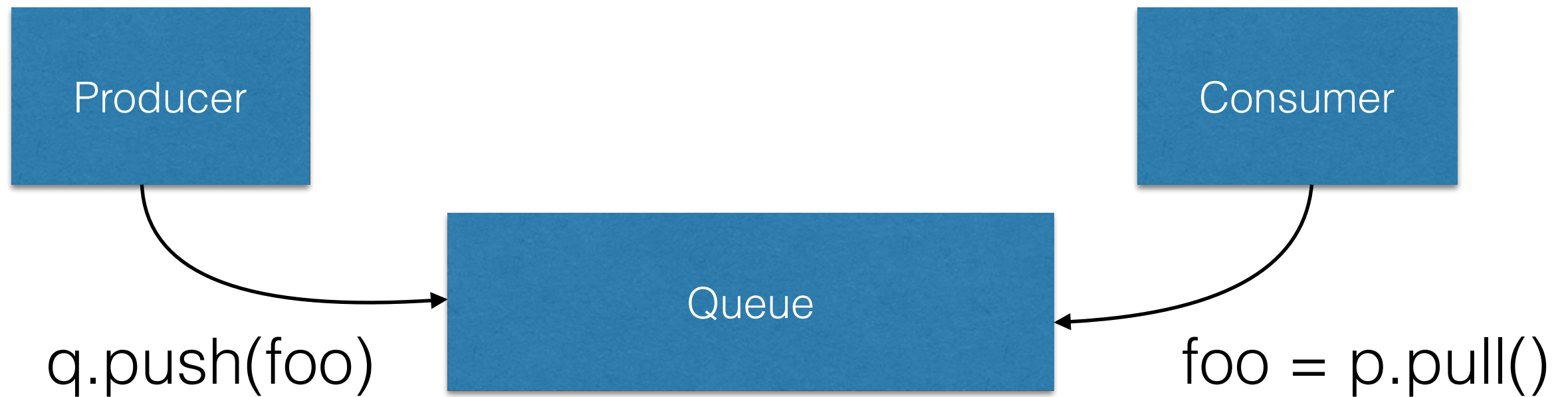
Zusammenfassung

- Verwendung von `new` verknüpft das erzeugte Objekt fest mit dem Erzeuger
 - Kopplung wird stärker (Erinnerung: low coupling)
 - Objekte lassen sich nicht mehr in Isolation betrachten
- Dependency Inversion trennt die Erzeugung und Verknüpfung von Objekten von ihrer Verwendung
- Implementierung von Komponenten kann ausgetauscht werden
- Das ist besonders nützlich bei Tests
- Und es erhöht die Wiederverwertbarkeit der Komponenten

Queues

- Dependency Inversion löst nicht das Problem der Kopplung zur Laufzeit
- Für den Methodenaufruf muss das Objekt vorhanden sein
- Wenn man eine noch losere Kopplung will, kann man eine Queue zwischen die Komponenten setzen





- Muss ein Consumer vorhanden sein, damit der Producer push ausführen kann?
- Muss ein Producer vorhanden sein, damit ein Consumer pull ausführen kann?
- Wieviele Producer und Consumer darf es geben?

Automatic Builds

Was will der Kunde?

Das Produkt

- Unser Produkt ist die Software, nicht der Source Code
- Wir müssen sicherstellen, dass wir aus dem Source Code die Software generieren können
- Unprofessionell: In Eclipse: Export jar
- Professionell: Automatisierung

Warum nicht von Hand?
Automatisierung erfordert
Aufwand!

Build Tools

- C,C++, alles was bash it: make
- Java
 - Apache Ant
 - Apache Maven
 - Gradle
- Man kann auch make benutzen um Java zu kompilieren (aber unter Windows kann es nervig sein)

Apache Ant

- Public release: 2000
- Imperative Beschreibung eines Builds
- Build wird in Targets unterteilt, z.B. class Files in ein Jar verpacken
- Targets können voneinander abhängen, z.B. um Klassen als Jar zu verpacken müssen sie erst kompiliert werden
- Zirkuläre Abhängigkeiten werden entdeckt

Hello ant

build.xml

```
<project>
  <target name="clean">
    <delete dir="build"/>
  </target>
  <target name="compile">
    <mkdir dir="build/classes"/>
    <javac srcdir="src" destdir="build/classes"/>
  </target>
  <target name="jar" depends="compile">
    <mkdir dir="build/jar"/>
    <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
      <manifest>
        <attribute name="Main-Class" value="HelloWorld"/>
      </manifest>
    </jar>
  </target>
  <target name="run" depends="clean,compile,jar">
    <java jar="build/jar/HelloWorld.jar" fork="true"/>
  </target>
</project>
```

Hello ant

```
<project>
  <target name="clean">
    <delete dir="build"/>
  </target>
  <target name="compile">
    <mkdir dir="build/classes"/>
    <javac srcdir="src" destdir="build/classes"/>
  </target>
  <target name="jar" depends="compile">
    <mkdir dir="build/jar"/>
    <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
      <manifest>
        <attribute name="Main-Class" value="HelloWorld"/>
      </manifest>
    </jar>
  </target>
  <target name="run" depends="clean,compile,jar">
    <java jar="build/jar/HelloWorld.jar" fork="true"/>
  </target>
</project>
```

Vor dem jar Target, erst compile ausführen



Vor/Nachteile

- Ant ist ziemlich flexibel
 - Man kann Logik abbilden
 - Man kann Erweiterungen als Plug-in (in Java) schreiben
- Ant ist keine "echte" Programmiersprache
- Bei komplexen Projekten oft nur Copy & Paste als Lösung möglich

Apache Maven

- Public release: 2004
- Rein deklarative Beschreibung eines Builds
 - Convention over configuration
 - Dependency resolution

Hello Maven

```
<project>
```

pom.xml

```
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>de.hhu.propra</groupId>
```

```
  <artifactId>hello</artifactId>
```

```
  <version>1.0</version>
```

```
  <properties>
```

```
    <project.build.sourceEncoding>
```

```
      UTF-8
```

```
    </project.build.sourceEncoding>
```

```
  </properties>
```

```
</project>
```

```

$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building hello 1.0
[INFO] -----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hello ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ hello ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/bendisposto/tmp/maven/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ hello ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/bendisposto/tmp/maven/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ hello ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ hello ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ hello ---
[INFO] Building jar: /Users/bendisposto/tmp/maven/target/hello-1.0.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.980 s
[INFO] Finished at: 2016-06-13T17:20:53+02:00
[INFO] Final Memory: 14M/81M
[INFO] -----

```


Konvention

- Es wird Java-Code kompiliert und ein Jar File generiert
- Der Java-Code liegt innerhalb des Projektordners unter `src/main/java`
- Die Tests liegen unter `/src/test/java`
(Achtung! Mit dieser pom.xml laufen die nicht)

Mit Test ...

<schnipp>

```
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ hello ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/bendisposto/tmp/maven/target/test-classes
[INFO] -----
[ERROR] COMPILATION ERROR :
[INFO] -----
[ERROR] /Users/bendisposto/tmp/maven/src/test/java/HelloTest.java:[1,24] package org.junit
does not exist
[ERROR] /Users/bendisposto/tmp/maven/src/test/java/HelloTest.java:[2,17] package org.junit
does not exist
[ERROR] /Users/bendisposto/tmp/maven/src/test/java/HelloTest.java:[6,4] cannot find symbol
  symbol:   class Test
  location: class HelloTest
[ERROR] /Users/bendisposto/tmp/maven/src/test/java/HelloTest.java:[8,5] cannot find symbol
  symbol:   method assertEquals(int,int)
  location: class HelloTest
[INFO] 4 errors
```

<schnipp>

Was fehlt?

Dependencies

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

```
$ mvn package
```

```
<snipp>
```

```
-----
T E S T S
-----
```

```
Running HelloTest
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.087 sec
```

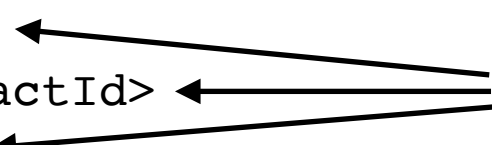
```
Results :
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

```
<snipp>
```

Dependencies

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```



Maven-Koordinaten

```
$ mvn package
```

```
<schnipp>
```

```
-----
T E S T S
-----
```

```
Running HelloTest
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.087 sec
```

```
Results :
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

```
<schnipp>
```

Maven Central

- Repository für Java Open Source Projekte
- <http://search.maven.org/>
- Über 260.000 Artefakte, 70 mio Downloads/Woche

Gradle

- Convention over Configuration
- Dependency Management (Maven kompatibel)
- Keine XML basierte Konfigurationsdatei
- Einbettung einer Build-Sprache in die Programmiersprache Groovy

Hello Gradle

```
apply plugin: "java"
```

```
build.gradle
```

```
$ gradle assemble  
:compileJava  
:processResources UP-TO-DATE  
:classes  
:jar  
:assemble
```

```
BUILD SUCCESSFUL
```


+ Tests

```
apply plugin: "java"                                build.gradle
```

```
repositories {  
    mavenCentral()  
}
```

```
dependencies {  
    testCompile "junit:junit:4.8.2"  
}
```

```
$ gradle build
:compileJava
:processResources UP-TO-DATE
:classes
:jar
:assemble
:compileTestJava
:processTestResources UP-TO-DATE
:testClasses
:test
:check
:build
```

BUILD SUCCESSFUL

Gradle

- build.gradle Datei
- Konventionen
 - Java Code in src/main/java (wie Maven)
 - Unit-Tests in src/test/java (wie Maven)
 - Ausgaben in build Verzeichnis

Best of both worlds

- Dependency Management kompatibel mit Maven
- Convention over Configuration
- Das Meiste geht deklarativ
- Aber eine echte Programmiersprache steht bereit

Beispiel: BBB

```
apply plugin: 'java'
apply plugin: 'eclipse'

repositories{
    mavenCentral()
}

dependencies {
    compile "javax.mail:mail:1.4"
    compile 'com.jayway.jsonpath:json-path:2.2.0'
    compile 'org.apache.derby:derby:10.12.1.1'

    testCompile "junit:junit:4.12"
}
```

Vier wichtige Plug-ins

- Java
- Application
- Eclipse
- Idea