LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME UND DATA MINING

Testing & Logging

SEP 2018

Tobias Lingelmann

2018-04-24

DBS

Wissenschaftliche Betreuer:
Daniel Kaltenthaler, Johannes Lohrer
Verantwortlicher Professor:
Prof. Dr. Peer Kröger

Inhalt

- Testing
  - Übersicht
  - JUnit
  - Test Driven Development
- Logging
  - Motivation
  - Funktionsweise
  - Frameworks

Testing

Zum Überprüfen, ob Testobjekt…

- … den Design-Anforderungen enstpricht.
- … korrekt auf alle (möglichen) Eingaben reagiert.
- … einigermaßen performant ist.
- … wie vorgesehen eingebettet werden kann.

- “Hilft beim Finden von Fehlern.”

Testing

- Funktionales Testen (Basierend auf Spezifikationen, Black-Box, „Was“)
- Strukturelles Testen (Überdeckungs-kritierien, White-Box, „Wie“)

Quelle

- Unit-Testing
- Integration-Testing
- System-Testing
- … → Softwaretechnik-Vorlesung

JUnit

- Framework zum Durchführen automatischer Unit-Tests
- Ermöglicht einfaches Testen kleiner Programmteile
- Test-Suits bestehend aus vielen kleinen Tests, möglich
- Liefert klar zu verstehendes Ergebnis: bestanden vs. fehlgeschlagen.
- Integriert in Eclipse ☺

JUnit

```
public class AccountTest {
     private Account testAccount;

     @Before
     public void setUp() {
          this.testAccount = new Account(100);
     }

     @After
     public void tearDown() {
          this.testAccount = null;
     }

     @Test
     public void testDeposit() {
          this.testAccount.deposit(10);
          assertEquals(110, this.testAccount.getBalance());
     }

     @Test
     public void testWithdraw() throws BalanceTooLowException {
          this.testAccount.withdraw(1);
          assertEquals(109, this.testAccount.getBalance());
     }

     @Test(expected = BalanceTooLowException.class)
     public void testOverdraft() throws BalanceTooLowException {
          this.testAccount.withdraw(110);
     }

}
```

JUnit

```
public class AccountTest {
     private Account testAccount;

     @Before
     public void setUp() {
          this.testAccount = new Account(100);
     }

     @After
     public void tearDown() {
          this.testAccount = null;
     }

     @Test
     public void testDeposit() {
          this.testAccount.deposit(10);
          assertEquals(110, this.testAccount.getBalance());
     }

     @Test
     public void testWithdraw() throws BalanceTooLowException {
          this.testAccount.withdraw(1);
          assertEquals(109, this.testAccount.getBalance());
     }

     @Test(expected = BalanceTooLowException.class)
     public void testOverdraft() throws BalanceTooLowException {
          this.testAccount.withdraw(110);
     }

}
```

Vor jedem Test

Nach jedem Test

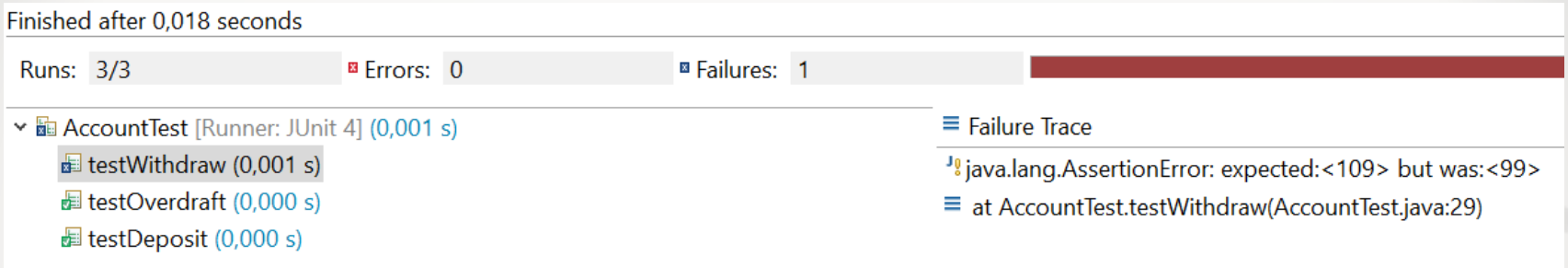
Test auf Gleichheit

Test auf Fehlerwurf

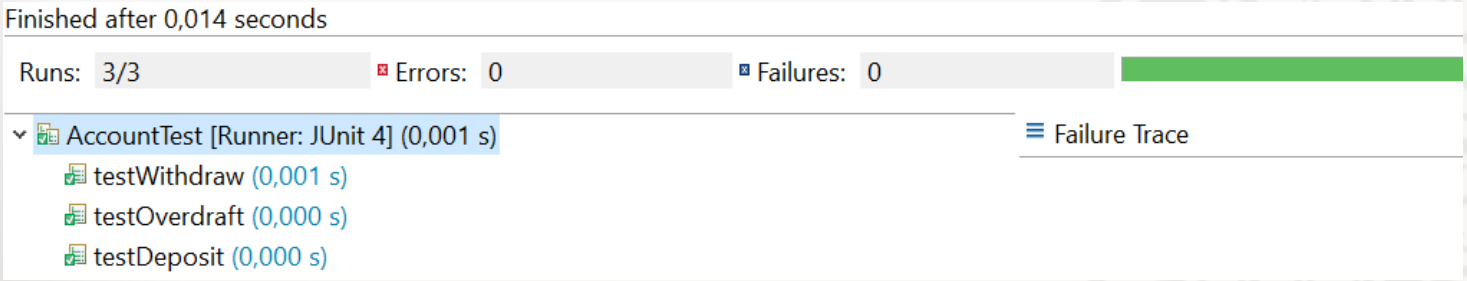
Auch: `@BeforeClass @AfterClass @Ignore`

JUnit

Ergebnis:



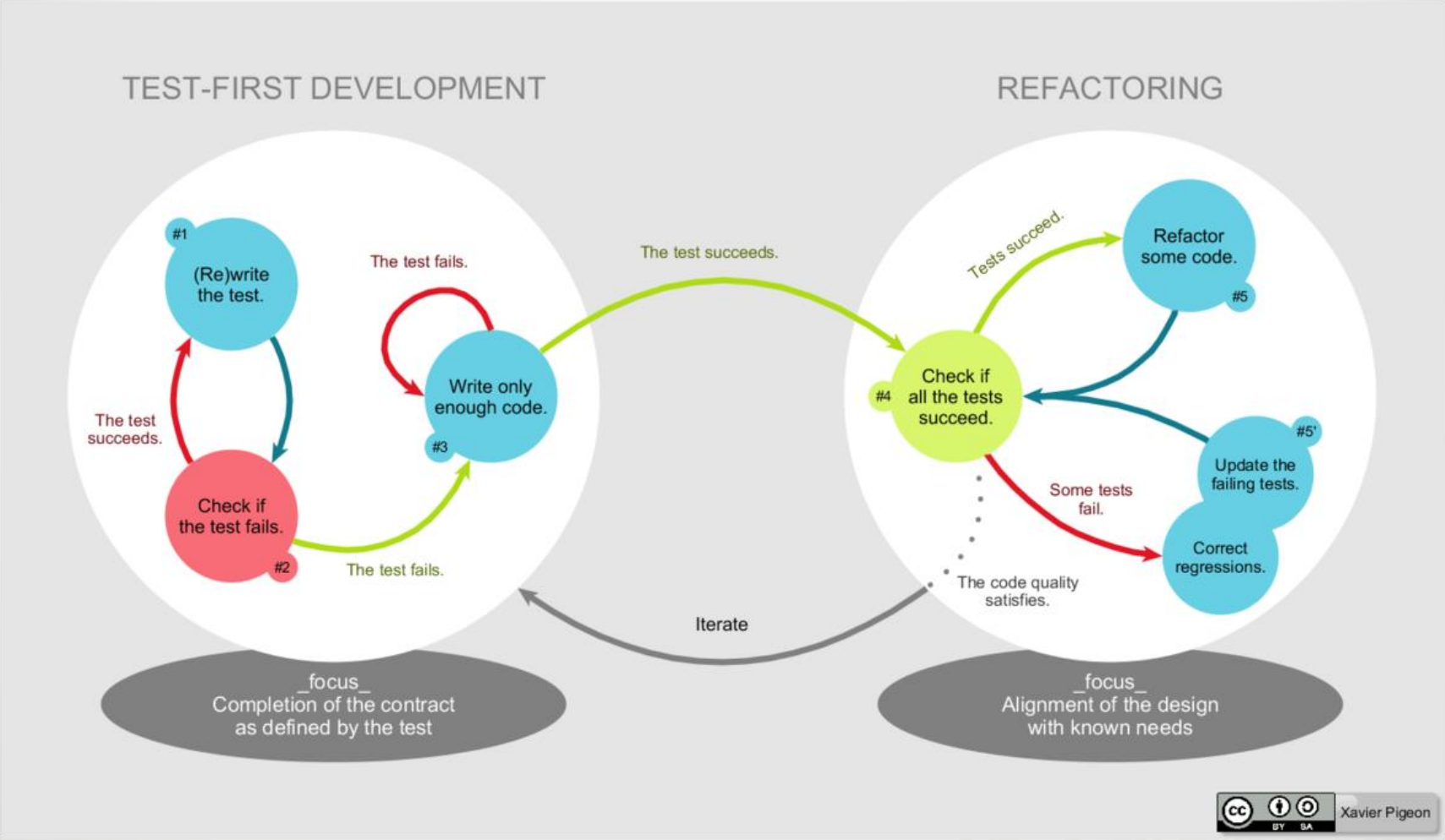
Mit korrektem Test:



JUnit

Assertion	Beschreibung
assertEquals	Testet ob zwei Werte gleich sind
assertTrue	Testet ob „wahr“
assertFalse	Testet ob „falsch“
assertNotNull	Testet ob Objekt nicht null ist
assertNull	Testet ob Objekt null ist
assertSame	Testet ob zwei Objekt-Referenzen gleich sind
assertNotSame	Testet ob zwei Objekt-Referenzen nicht gleich sind
assertArrayEquals	Testet ob zwei Arrays gleich sind
fail	Schlägt fehl

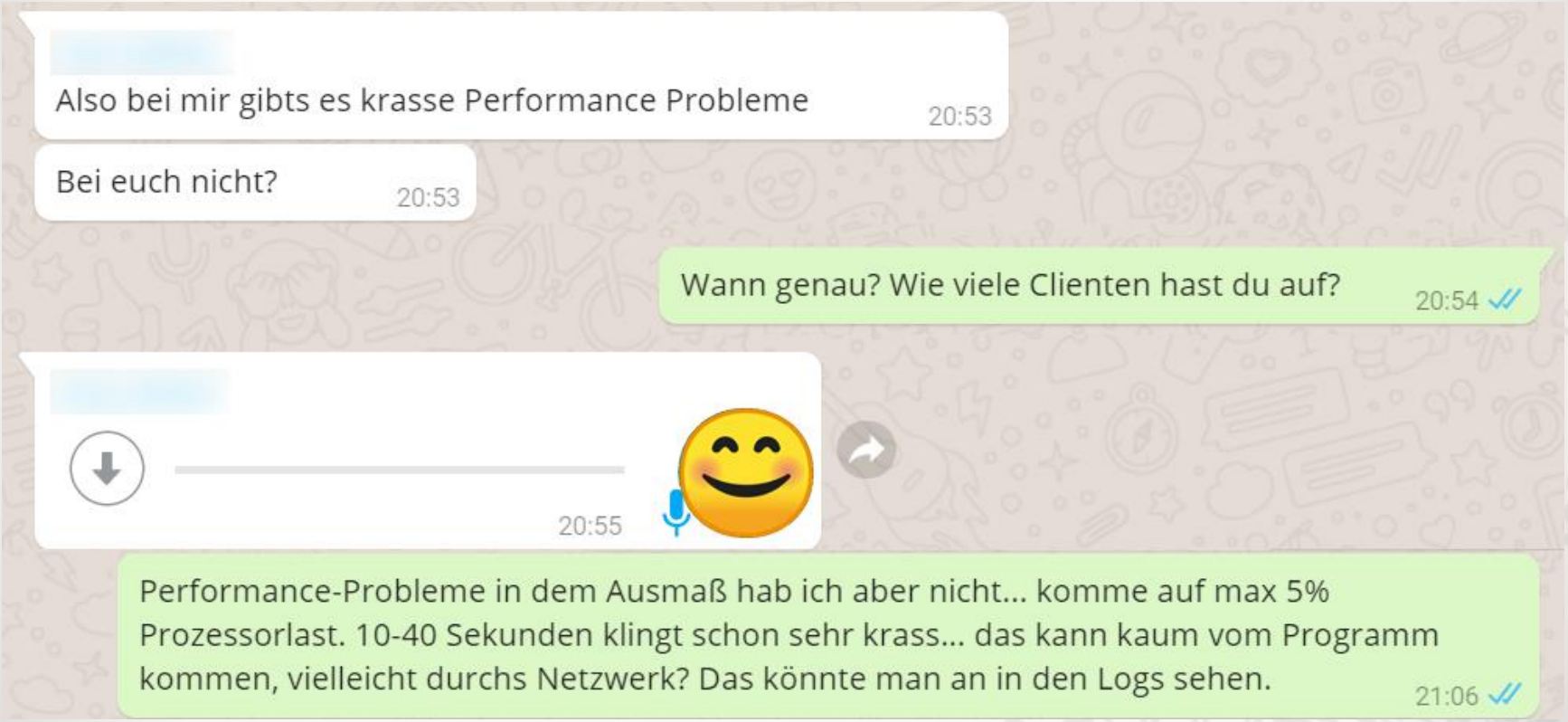
Test Driven Development



Logging

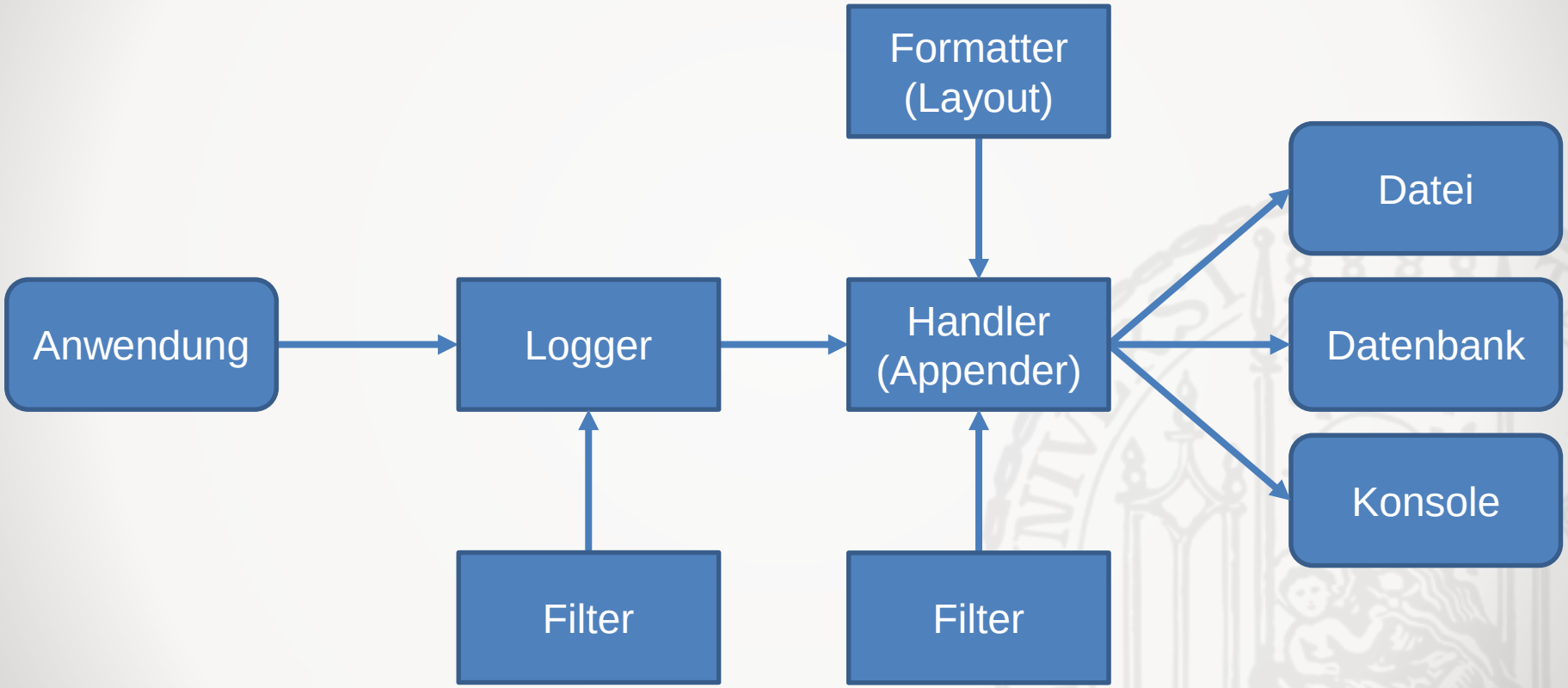
- Automatisches Erstellen eines Protokolls
- Aufzeichnung zur Nachvollziehbarkeit von (Fehler-)Zuständen
- Standardisierte Aufzeichnung ermöglicht (automatische) Auswertung
- Ausgabe auf der Konsole, in Dateien, in Datenbanken, übers Netzwerk, …

Logging


Also bei mir gibts es krasse Performance Probleme
20:53
Bei euch nicht?
20:53
Wann genau? Wie viele Clienten hast du auf?
20:54
20:55
Performance-Probleme in dem Ausmaß hab ich aber nicht... komme auf max 5% Prozessorlast. 10-40 Sekunden klingt schon sehr krass... das kann kaum vom Programm kommen, vielleicht durchs Netzwerk? Das könnte man an in den Logs sehen.
21:06

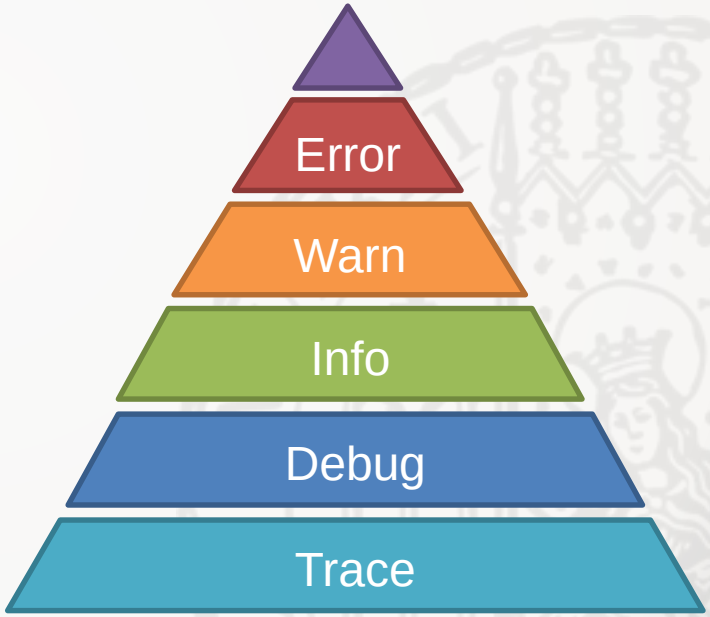
Logging

Aufbau



Logging

- Hierarchische „Level“:
  - java.util.logging:
    - Severe
    - Warning
    - Info
    - Config
    - Fine
    - Finer
    - Finest
  - Log4j2:
    - Fatal
    - Error
    - Warn
    - Info
    - Debug
    - Trace
    - All



Logging

```
import java.util.logging.*;

public class Nose {
     private static Logger logger = Logger.getLogger("de.test.logtest");
     private static ConsoleHandler ch = new ConsoleHandler();

     public static void main(String argv[]) {
          // Avoid forwarding logs to parent's handler
          logger.setUseParentHandlers(false);
          // Add own handler
          logger.addHandler(ch);
          // Set level for handler
          ch.setLevel(Level.FINER);
          // Set level for logger
          logger.setLevel(Level.FINER);
          // Log a simple INFO message.
          logger.info("doing stuff");
          try {
             sneeze();
          } catch (Exception ex) {
             logger.log(Level.WARNING, "trouble sneezing", ex);
          }
          logger.fine("done");
     }

     public static void sneeze() throws Exception {
          throw new Exception();
     }
}
```

Logging

Nose4j.java

```java
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Nose4j {
	private static Logger logger = LogManager.getLogger("de.test.logtest");

	public static void main(String argv[]) {
		logger.info("doing stuff");
		try {
		   sneeze();
		} catch (Exception ex) {
		   logger.catching(ex);
		}
		logger.debug("done");
	}

	public static void sneeze() throws Exception {
		throw new Exception();
	}
}
```

log4j2.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
	<Appenders>
		<Console name="Console" target="SYSTEM_OUT">
			<PatternLayout pattern="%d{HH:mm:ss.SSS} [%t %T] %-5level %c{1.} - %msg%n" />
		</Console>
	</Appenders>
	<Loggers>
		<Root level="trace" includeLocation="false">
			<AppenderRef ref="Console" />
		</Root>
	</Loggers>
</Configuration>
```

Linksammlung

- https://junit.org/junit5/docs/current/api/
- https://www.tutorialspoint.com/junit/index.htm
- https://en.wikipedia.org/wiki/Code_coverage
- https://en.wikipedia.org/wiki/Software_testing
- http://blog.codepipes.com/testing/software-testing-antipatterns.html
- https://docs.oracle.com/javase/9/docs/api/java/util/logging/package-summary.html
- https://logging.apache.org/log4j/2.x/manual/index.html