Testing & Logging

- JUnit
- Test Driven Development
- Logging

JUnit

- Testen Allgemein:

  Verhält sich Programm wie erwartet?

  Manuell / Automatisiert

  Auf verschiedenen Ebenen

  (Sytem > … > Klasse > Methode)

- JUnit

  Java-Framework zum Durchführen automatisierter UnitTests

  Testen kleinerer Programmteile in Isolation von anderen Programmteilen

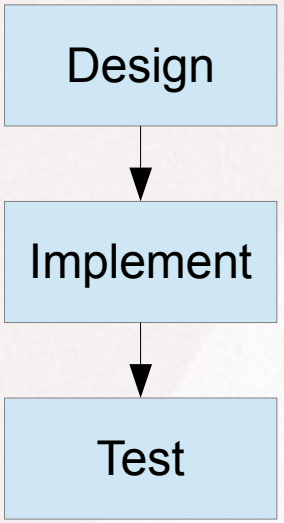
  Ergebnis bestanden / nicht bestanden

Beispiel

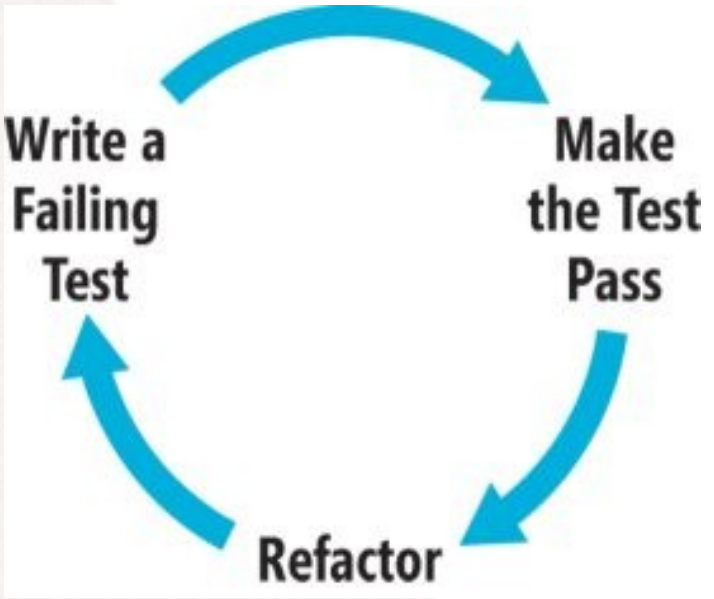
```
public class AccountTest {
      private Account a;

      @Before
      public void setUp() {
            a = new Account(„Customer“);
            a.deposit(100);
      }

      @Test
      public void testDeposit() {
            assertEquals(100, a.getBalance());
      }

      @After
      public void tearDown() {
            a = null;
      }
```

Assertion	Funktionsweise
fail	Schlägt fehl
assertTrue	Überprüft ob die Bedingung wahr ist
assertEquals	Überprüft ob die Objekte gleich sind (mit equals)
assertNotSame	Überprüft dass die Referenzen auf unterschiedliche Objekte zeigen (unabhängig von equals)
assertNull	Überprüft dass Parameter null ist
assertNotNull	Überprüft dass Parameter nicht null ist
assertArrayEquals	Überprüft ob zwei Arrays gleich sind

Test Driven Development



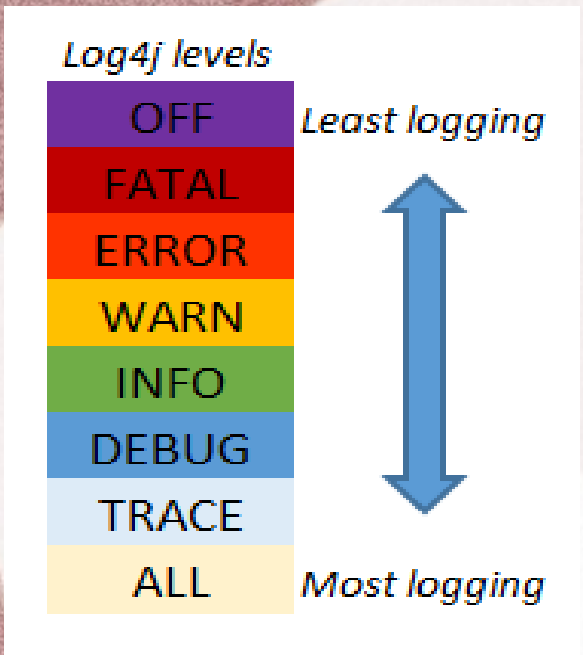
VS



Erst Test erstellen, dann den entsprechenden Code

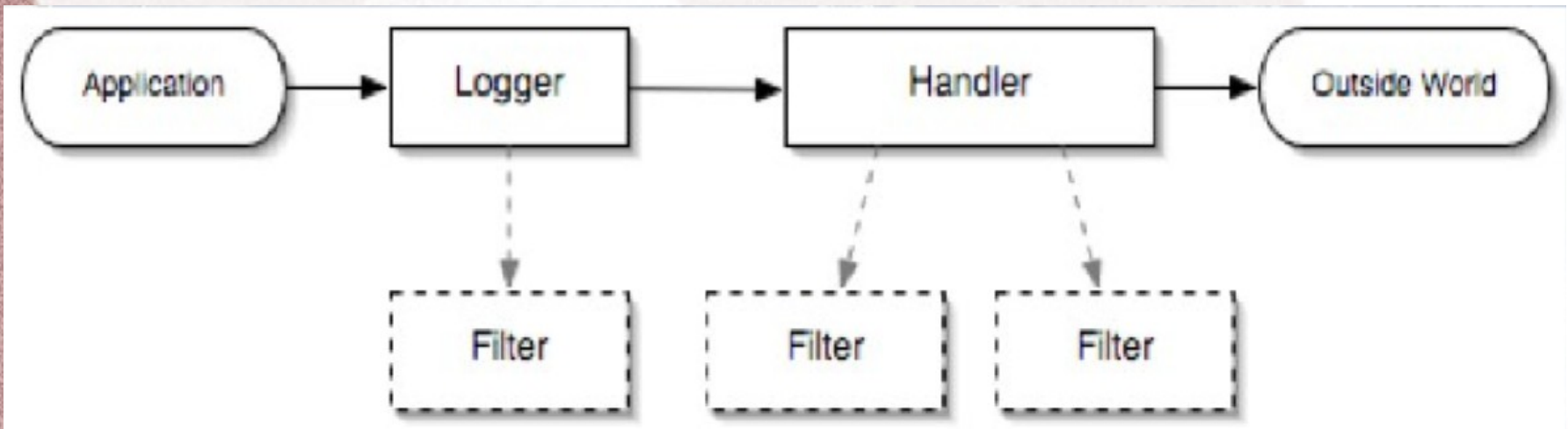
Logging

- Protokollieren von Programmzuständen
- Standardisiert / Auswertbar
- Ausgabe in Console, Datei, Netzwerk...
- JUL vs log4j



Output in File LogA.txt:

```
<?xml version="1.0" encoding="windows-1252" standalone="no"?>

  <!DOCTYPE log SYSTEM "logger.dtd">
  <log>
  <record>
      <date>2016-11-23 20:53:38 </date>
      <millis>1039636418456</millis>
      <sequence>1</sequence>
       <logger>LogA</logger>
      <level>WARNING</level>
      <class>LogA</class>
      <method>doA</method>
      <thread>10</thread>
      <message>Error in LogA!</message>
   </record>
</log>
```



```java
public class Log4jDemo
{
  private final static Logger log = Logger.getLogger( Log4jDemo.class );

  public static void main( String[] args )
  {
    BasicConfigurator.configure();

    log.info( "Dann mal los." );

    try
    {
      ((Object) null).toString();
    }
    catch ( Exception e )
    {
      log.error( "oh oh", e );
    }

    log.info( "Ging alles glatt." );
  }
}
```

Die Ausgabe ist:

```
0 [main] INFO com.tutego.insel.logging.Log4jDemo  - Dann mal los.
2 [main] ERROR com.tutego.insel.logging.Log4jDemo  - oh oh
java.lang.NullPointerException
    at com.tutego.insel.logging.Log4jDemo.main(Log4jDemo.java:18)
2 [main] INFO com.tutego.insel.logging.Log4jDemo  - Ging alles glatt.
```

```java
public class logA {
    static Logger logger;
    Handler fh;
    Formatter xml;

    public logA() throws IOException {
        logger = Logger.getLogger("LogA");
        fh = new FileHandler("LogA.xml");
        xml = new XMLFormatter();

        fh.setFormatter(xml);
        logger.addHandler(fh);
    }
    public void doA() {
        double r = Math.random();

        if(r<0.5){ logger.warning("Error in LogA!");
        }else{
        logger.info("LogA ok :)");
            }
    }
}
```

Je nachdem, was passiert, wird Nachricht mit LogLevel „warning“ oder die mit LogLevel „info“ ausgegeben