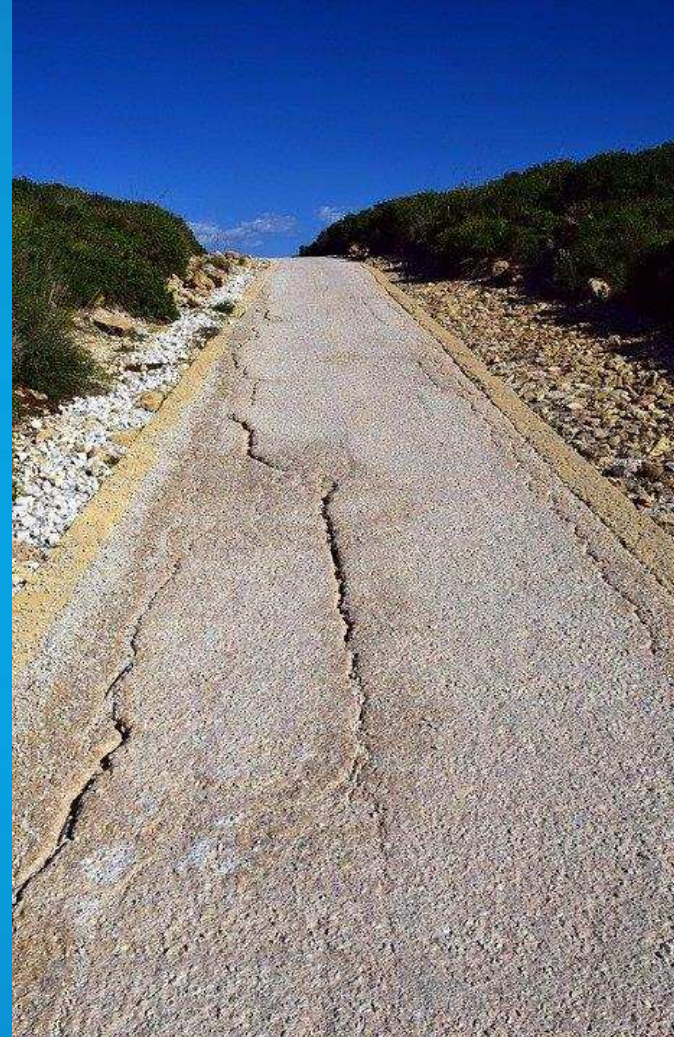


# Journey to object calisthenics

Urs Blum



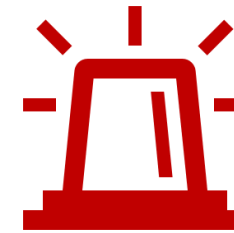
# Idea

- Find some code and apply the object calisthenics rules to it

```
@Override
public CalculateVerarbeitungsartResult calculate() throws BSEException {
    LOGGER.debug("calculateAndModifyVerarbeitungsart with DATA: " + data);

    CalculateVerarbeitungsartResult result = super.calculate();
    if (Objects.nonNull(result)) {
        return result;
    }
    if (isTaetigkeitJsonEmpty(taetigkeit)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0347_MANUELLE_BEARBEITUNG);
    }

    if (isManuelleAuslenkung(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0347_MANUELLE_BEARBEITUNG);
    }
    else if (isUnerlaubteFranchiseReduktion()) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0389_UNERLAUBTE_FRANCHISENREDUKTION);
    }
    else if (isRueckwirkendeMutation(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0349_RUECKWIRKENDE_POLMUT);
    }
    else if (isHausarztWechsel60TageInVergangenheit(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0350_HAUSARZT_WECHSEL_VERGANGENHEIT);
    }
    else if (isUnfallEinschluss60TageInVergangenheit(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0340_UNFALLRISIKO_60_TAGE_VERGANGENHEIT);
    }
    else if (isUnfallausschlussInVergangenheit(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0341_UNFALLRISIKO_IN_VERGANGENHEIT);
    }
    else if (isHausarztInZukunftBeendet()) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0351_HAUSARZT_WECHSEL_BEENDETER_HAUSARZT);
    }
    else if (isSpezielleUnterjaehrigeAenderung()) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0352_UNTERJAEHRIGE_VERSICHERUNGSAENDERUNG);
    }
    else if (isUvgVersichert5JahreInZukunft(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterLeiten(E0343_UNGVERSICHERTBIS_MEHR_5JAHRE_ZUKUNFT);
    }
    }
    return CalculateVerarbeitungsartResult.createAutomatisch();
}
```



Before refactoring, i should check if we have some test...

# Look at the existing Tests

```
CalculateVerarbeitungsartResult result = calculator.calculate();

assertNull(result.getAuslenkungsgrund());
}

@Test
public void unterjaehrigeAenderungVonNichtStandardDKPZuStandardDKP() throws BSEException {
    taetigkeit = createTaetigkeit(id: "1-2346TR", json: {}, partnerNo: "33333333", uNo: "1-2358972357", verarbeitungsart: "03", kundenrelevantInfo: false);
    data = createGrundversicherungData(LocalDate.of(LocalDate.now().plusYears(1).getYear(), Month.MARCH, dayOfMonth: 1), VersichertesRisiko.RISIKO_KRANKHEIT,
        VersichertesRisiko.RISIKO_KRANKHEIT, uvjVersicherterAlt: null, uvjVersicherterNeu: null, franchiseBetragAlt: 500, franchiseBetragNeu: 500, franchiseStufeAlt: 4, franchiseStufeNeu: 4, Anbieter.CSS, Anbieter.CSS, CALLMED.getValue(),
        DPK_CSS_ARC_AUX.getValue(), leistungsbringerGruppenNrAlt: "12345678", leistungsbringerGruppenNrNeu: "12345678", leistungsbringerNrAlt: "11111111", leistungsbringerNrNeu: "11111111", manuellAuslenken: false);

    calculator = new GrundversicherungAuslenkungsgrundCalculator(taetigkeit, data, accessor);
    CalculateVerarbeitungsartResult result = calculator.calculate();

    assertEquals( @expected: "E0352_Unterjaehrige_Versicherungsänderung", result.getAuslenkungsgrund().getValue());
}

@Test
public void unterjaehrigeAenderungVonNichtStandardDKPZuNichtStandardDKP() throws BSEException {
    taetigkeit = createTaetigkeit(id: "1-2346TR", json: {}, partnerNo: "33333333", uNo: "1-2358972357", verarbeitungsart: "03", kundenrelevantInfo: false);
    data = createGrundversicherungData(LocalDate.of(LocalDate.now().plusYears(1).getYear(), Month.MARCH, dayOfMonth: 1), VersichertesRisiko.RISIKO_KRANKHEIT,
        VersichertesRisiko.RISIKO_KRANKHEIT, uvjVersicherterAlt: null, uvjVersicherterNeu: null, franchiseBetragAlt: 500, franchiseBetragNeu: 500, franchiseStufeAlt: 4, franchiseStufeNeu: 4, Anbieter.CSS, Anbieter.CSS, CALLMED.getValue(),
        leistungsbringerGruppenNrAlt: "12345678", leistungsbringerGruppenNrNeu: "12345678", leistungsbringerNrAlt: "11111111", leistungsbringerNrNeu: "11111111", manuellAuslenken: false);

    calculator = new GrundversicherungAuslenkungsgrundCalculator(taetigkeit, data, accessor);
    CalculateVerarbeitungsartResult result = calculator.calculate();

    assertNull(result.getAuslenkungsgrund());
}

@Test
public void unterjaehrigeAenderungVonNichtStandardDKPZuNichtStandardDKPAndererFranchise() throws BSEException {
    taetigkeit = createTaetigkeit(id: "1-2346TR", json: {}, partnerNo: "33333333", uNo: "1-2358972357", verarbeitungsart: "03", kundenrelevantInfo: false);
    data = createGrundversicherungData(LocalDate.of(LocalDate.now().plusYears(1).getYear(), Month.MARCH, dayOfMonth: 1), VersichertesRisiko.RISIKO_KRANKHEIT,
        VersichertesRisiko.RISIKO_KRANKHEIT, uvjVersicherterAlt: null, uvjVersicherterNeu: null, franchiseBetragAlt: 500, franchiseBetragNeu: 1000, franchiseStufeAlt: 4, franchiseStufeNeu: 6, Anbieter.CSS, Anbieter.CSS, CALLMED.getValue(),
        leistungsbringerGruppenNrAlt: "12345678", leistungsbringerGruppenNrNeu: "12345678", leistungsbringerNrAlt: "11111111", leistungsbringerNrNeu: "11111111", manuellAuslenken: false);

    calculator = new GrundversicherungAuslenkungsgrundCalculator(taetigkeit, data, accessor);
    CalculateVerarbeitungsartResult result = calculator.calculate();

    assertEquals( @expected: "E0352_Unterjaehrige_Versicherungsänderung", result.getAuslenkungsgrund().getValue());
}

@Test
public void unterjaehrigeAenderungVonNichtStandardDKPZuNichtStandardDKPAndererAnbieter() throws BSEException {
    taetigkeit = createTaetigkeit(id: "1-2346TR", json: {}, partnerNo: "33333333", uNo: "1-2358972357", verarbeitungsart: "03", kundenrelevantInfo: false);
    data = createGrundversicherungData(LocalDate.of(LocalDate.now().plusYears(1).getYear(), Month.MARCH, dayOfMonth: 1), VersichertesRisiko.RISIKO_KRANKHEIT,
        VersichertesRisiko.RISIKO_KRANKHEIT, uvjVersicherterAlt: null, uvjVersicherterNeu: null, franchiseBetragAlt: 500, franchiseBetragNeu: 1000, franchiseStufeAlt: 4, franchiseStufeNeu: 6, Anbieter.CSS, Anbieter.ARCOSANA, CALLMED.getValue(),
        MULTIMED.getValue(), leistungsbringerGruppenNrAlt: "12345678", leistungsbringerGruppenNrNeu: "12345678", leistungsbringerNrAlt: "11111111", leistungsbringerNrNeu: "11111111", manuellAuslenken: false);

    calculator = new GrundversicherungAuslenkungsgrundCalculator(taetigkeit, data, accessor);
    CalculateVerarbeitungsartResult result = calculator.calculate();

    assertEquals( @expected: "E0352_Unterjaehrige_Versicherungsänderung", result.getAuslenkungsgrund().getValue());
}
```



Nice Tests (Arrange, Act, Assert)



They look pretty much all similar  
-> Let's refactor them to  
Parameterized Tests

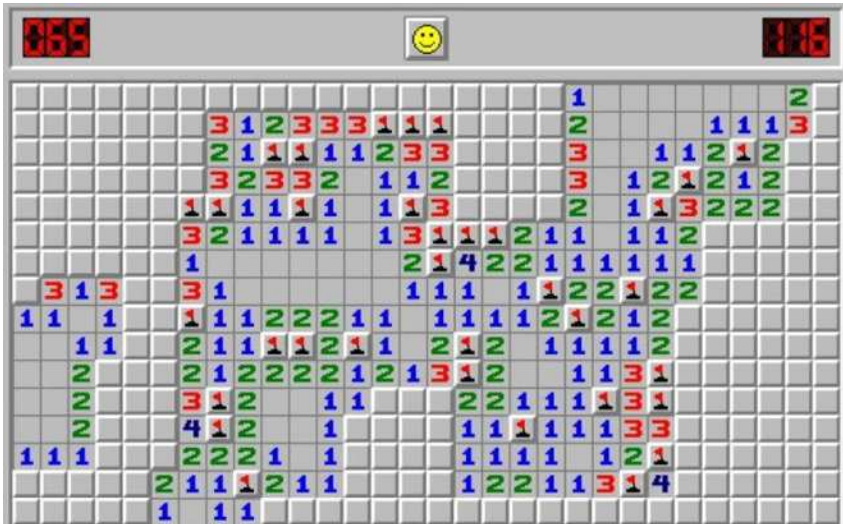


Oh, only Junit 4 -> Then first  
migrate to Junit 5



# What a silly idea to migrate to Junit 5 just to remove some elses...

- Pitfalls



- You need a whole bunch of new dependencies, including dependencies you only need at Runtime (junit-vintage-engine)
- Mockito throws an error when there is an unused mock (unless you use lenient)
- The order in Assertions has changed, the optional message is now at the end
- @Ignore does compile, but has no effect -> @Disabled
- Lot of search-replace (@Before -> @BeforeEach, @RunWith -> @ExtendWith, Imports, ...)



Powermock is not supported



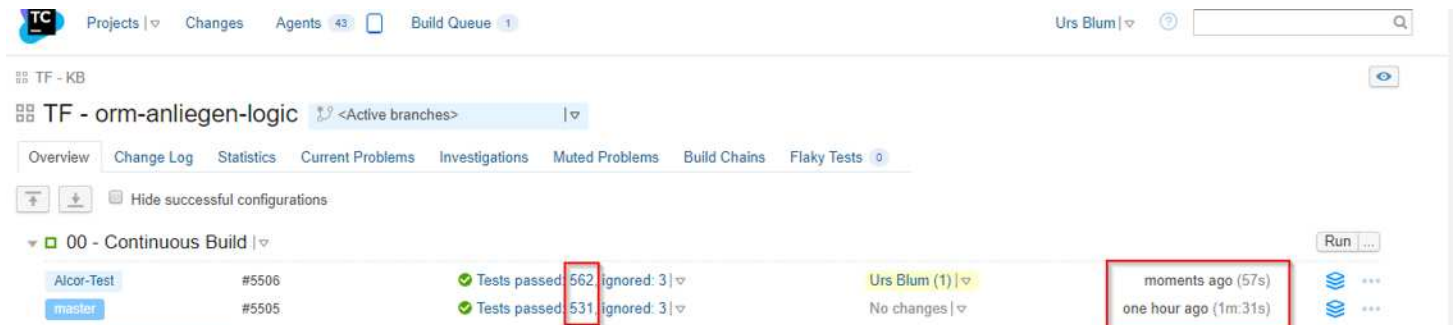
MockStatic as main-reason for powermock is not supported

- ...

# Result of Migration



## Sideeffects:



TC Projects | Changes Agents 43 | Build Queue 1 Urs Blum |

TF - KB

TF - orm-anliegen-logic <Active branches>

Overview Change Log Statistics Current Problems Investigations Muted Problems Build Chains Flaky Tests 0

Hide successful configurations

00 - Continuous Build

Alcor-Test	#5506	Tests passed: 562, ignored: 3	Urs Blum (1)	moments ago (57s)
master	#5505	Tests passed: 531, ignored: 3	No changes	one hour ago (1m:31s)

# Parameterized Test and no Elses

```
Mockito.lenient().when(accessor.getLeistungserbringerQueryAccessor()).thenReturn(leistungserbringerQueryAccessor);
}

@ParameterizedTest
@MethodSource("provideTestData")
public void shouldReturnExpectedAuslenkungsgrund(TaetigkeitVerarbeitung taetigkeitVerarbeitung, GrundversicherungData grundversicherungData,
String expectedAuslenkungsgrund) throws BSEException {
    GrundversicherungAuslenkungsgrundCalculator calculator = new GrundversicherungAuslenkungsgrundCalculator(taetigkeitVerarbeitung,
grundversicherungData, accessor);

    CalculateVerarbeitungsartResult result = calculator.calculate();

    assertEquals(expectedAuslenkungsgrund, result.getAuslenkungsgrund() != null ? result.getAuslenkungsgrund().getValue() : null);
}

@Test
public void unerlaubteFranchiseReduktion() throws BSEException {
    LocalDate firstDecember = LocalDate.of(year, 2020, Month.DECEMBER, dayOfMonth, 1);

    taetigkeit = createTaetigkeit(id: "1-234GTR", json: "{}", postnr: "33333333", idnr: "1-2358972357", verarbeitungsart: "03", kundenrelevanz: false);
}

GrundversicherungAuslenkungsgrundCalculatorTest.setUp()
```

Tests passed: 22 of 22 tests - 1 s 649 ms

Test Results	Time
GrundversicherungAuslenkungsgrundCalculatorTest	1 s 649 ms
unerlaubteFranchiseReduktion()	1 s 565 ms
unfallabschlussUnterjaehrig()	3 ms
shouldReturnExpectedAuslenkungsgrund(TaetigkeitVerarbeitung, GrundversicherungData, String expectedAuslenkungsgrund)	75 ms
(1) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=1)	35 ms
(2) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=1)	3 ms
(3) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(4) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(5) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(6) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(7) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(8) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(9) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(10) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(11) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=2)	2 ms
(12) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	3 ms
(13) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	2 ms
(14) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms
(15) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms
(16) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms
(17) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms
(18) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms
(19) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms
(20) TaetigkeitVerarbeitung(id="1-234GTR", taetigkeitNr="null", jsonDa=3)	1 ms

```
@Override
public CalculateVerarbeitungsartResult calculate() throws BSEException {
    LOGGER.debug("calculateAndModifyVerarbeitungsart with DATA: " + data);

    CalculateVerarbeitungsartResult result = super.calculate();
    if (Objects.nonNull(result)) {
        return result;
    }
    if (isTaetigkeitJsonEmpty(taetigkeit)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0347_MANUELLE_BEARBEITUNG);
    }
    if (isManuelleAuslenkung(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0347_MANUELLE_BEARBEITUNG);
    }
    if (isUnerlaubteFranchiseReduktion()) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0389_UNERLAUBTE_FRANCHISENREDUKTION);
    }
    if (isRueckwirkendeMutation(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0349_RUECKWIRKENDE_POLMUT);
    }
    if (isHausarztWechsel60TageInVergangenheit(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0350_HAUSARZT_WECHSEL_VERGANGENHEIT);
    }
    if (isUnfallabschluss60TageInVergangenheit(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0340_UNFALLRISIKO_60_TAGE_VERGANGENHEIT);
    }
    if (isUnfallabschlussInVergangenheit(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0341_UNFALLRISIKO_IN_VERGANGENHEIT);
    }
    if (isHausarztInZukunftBeendet()) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0351_HAUSARZT_WECHSEL_BEENDETER_HAUSARZT);
    }
    if (isSpezielleUnterjaehrigeAenderung()) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0352_UNTERJAEHRIGE_VERSICHERUNGSANDENDERUNG);
    }
    if (isUvgVersichert5JahreInZukunft(this.data)) {
        return CalculateVerarbeitungsartResult.createWeiterleiten(E0343_UVGVERSICHERTBIS_MEHR_5JAHRE_ZUKUNFT);
    }
    return CalculateVerarbeitungsartResult.createAutomatisch();
}
```