LU09.A02 - Refactoring (Zusatz)

Verbessern Sie ein Programm nach den Clean-Code-Prinzipien, ohne sein Verhalten zu ändern.
Zeit: ca. 30 Minuten · **Sozialform:** Einzelarbeit · **Zusatzaufgabe** (Advanced), keine Abgabe

Theorie: LU09b - Clean Code

Ausgangslage

Die Funktion `report` erstellt ein Zeugnis mit drei Fächern. Sie funktioniert, und pylint gibt ihr trotzdem die Note **9.35/10**. Lesbar ist sie nicht.

Erstellen Sie in PyCharm ein neues Projekt mit diesen zwei Dateien:

report.py

```
def report(n, m, d, e):
    # Mathe
    s = 0
    for g in m:
        s = s + g
    a1 = round(s / len(m) * 2) / 2
    # Deutsch
    s = 0
    for g in d:
        s = s + g
    a2 = round(s / len(d) * 2) / 2
    # Englisch
    s = 0
    for g in e:
        s = s + g
    a3 = round(s / len(e) * 2) / 2
    x = round((a1 + a2 + a3) / 3, 1)
    t = 'Zeugnis ' + n + '\n'
    t = t + 'Mathe: ' + str(a1) + '\n'
    t = t + 'Deutsch: ' + str(a2) + '\n'
    t = t + 'Englisch: ' + str(a3) + '\n'
    t = t + 'Schnitt: ' + str(x) + '\n'
    c = 0
    if a1 < 4:
        c = c + 1
    if a2 < 4:
        c = c + 1
    if a3 < 4:
```

```
        c = c + 1
    if x >= 4 and c <= 1:
        t = t + 'promoviert'
    else:
        t = t + 'nicht promoviert'
    print(t)
    return t
```

test_report.py

```
""" Unit Tests für report – sie müssen vor und nach dem Refactoring
grün sein """
from report import report


def test_promoted():
    """ alle Fächer genügend """
    assert report('Mia', [4.5, 5.0], [5.5, 6.0], [4.5, 4.5]) == (
        'Zeugnis Mia\n'
        'Mathe: 5.0\n'
        'Deutsch: 6.0\n'
        'Englisch: 4.5\n'
        'Schnitt: 5.2\n'
        'promoviert'
    )


def test_one_fail_promoted():
    """ eine ungenügende Note ist erlaubt, wenn der Schnitt reicht """
    assert report('Tim', [3.0, 3.5], [5.0, 5.0], [4.5,
4.5]).endswith('Schnitt: 4.2\npromoviert')


def test_two_fails_not_promoted():
    """ zwei ungenügende Noten führen zur Nichtpromotion """
    assert report('Ela', [3.0, 3.5], [3.5, 3.5], [6.0,
6.0]).endswith('nicht promoviert')


def test_average_too_low():
    """ Schnitt unter 4 führt zur Nichtpromotion """
    assert report('Leo', [3.5], [4.0], [4.0]).endswith('Schnitt:
3.8\nnicht promoviert')
```

Auftrag

1. Code Smells finden

Führen Sie die Tests mit `pytest -v` aus: Alle vier sind grün. Notieren Sie danach mindestens **fünf** Code Smells in `report.py` und ordnen Sie jedem das verletzte Prinzip zu (sprechende Namen, SRP, DRY, KISS).

2. Refactoring in kleinen Schritten

Verbessern Sie den Code Schritt für Schritt. Führen Sie **nach jedem Schritt** die Tests aus.

- Geben Sie Parametern und Variablen sprechende Namen.
- Lösen Sie die wiederholte Berechnung in eine Funktion heraus.
- Ersetzen Sie Magic Numbers durch Konstanten.
- Trennen Sie Berechnung, Promotionsentscheid und Ausgabe. Die Funktion `report` soll nichts mehr mit `print` ausgeben.

Der Funktionsname `report` und die Reihenfolge der Parameter bleiben gleich, sonst passen die Tests nicht mehr.

3. Prüfen

- Alle Tests sind grün.
- pylint meldet `10.00/10`.
- Schreiben Sie mindestens einen zusätzlichen Test für eine der neuen Hilfsfunktionen. Was fällt Ihnen auf, wenn Sie das mit der alten Version vergleichen?

M450-LU09, M450-D1B, M450-D1A

Marcel Suter

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu09/aufgaben/refactoring?rev=1795421967

Last update: **2026/11/23 09:19**