# Übungsaufgabe 2 (Sommer 2022)

Der Energieversorger Wind und Sonne AG möchte den Prozess zur Strom-Abrechnung weiter digitalisieren. Innerhalb dieses Projekts sollen mehrere Apps entwickelt werden. Dabei sollen Sie bei der Planung, Umsetzung und Einführung mitarbeiten und unterstützen.

Zum Testen der Methoden der zu entwickelnden Software kommen Unit-Tests zum Einsatz.

a) Beschreiben Sie, was ein Unit-Test ist und wozu er verwendet wird.

ein unit test testet die kleinstmögliche einheit vom code
(normalerweise immer eine einzelne Funktion)

Man verwendet diesen um die funktionalität dieser Funktion isoliert vom rest sicherzustellen

b) Nennen und beschreiben Sie zwei Eigenschaften, welche ein Unit-Test erfüllen sollte.

1. Isolation
2. Wiederholbarkeit

Im Rahmen des Projektes wurde eine Methode entwickelt, welche folgende funktionalen Anforderungen erfüllen soll:

- Die Methode soll die positive Differenz zwischen einem alten und einem neuen Zählerstand, welche als Parameter übergeben werden, berechnen und diesen Wert zurückgeben.
- Die Differenz soll nur dann berechnet werden, wenn die übergebenen Werte nicht negativ sind und der alte Zählerstand nicht größer als der neue Zählerstand ist. Ansonsten soll die Methode den Wert -1 zurückgeben.

Quellcode der Funktion:

```
int berechneDifferenz (int zaehlerstandAlt, int zaehlerstandNeu) {
    int differenz = -1;
    if (zaehlerstandAlt >= 0 && zaehlerstandNeu > 0) {
        if (zaehlerstandAlt < zaehlerstandNeu) {
            differenz = zaehlerstandNeu - zaehlerstandAlt;
        }
    }
    return differenz;
}
```

c) Zum Testen der Methode wurde folgender Unittest entwickelt.

```
@Test
void testeBerechneZahlungsdifferenz () {
     assertEquals(20, berechneDifferenz(240, 260));     // Testfall 1
     assertEquals(0, berechneDifferenz(120, 120));      // Testfall 2
     assertEquals(0, berechneDifferenz(0, 0));          // Testfall 3
     assertEquals(80, berechneDifferenz(0, 80));        // Testfall 4
     assertEquals(-1, berechneDifferenz(440, 420));     // Testfall 5
}
```

| | |
|---|---|
| `assertEquals(Expected, Actual)` | Prüft, ob die Werte für `Actual` und `Expected` gleich sind. Ist dieses nicht der Fall, schlägt der Test fehl. |

Überprüfen Sie, ob der Unit-Test erfolgreich ist oder fehlschlägt. Tragen Sie dazu das Ergebnis von jedem Testfall in die Tabelle ein. (OK -wenn der Test erfolgreich ist, Fehler - wenn der Test fehlschlägt.)

| Testfall | Ergebnis des Tests |
|---|---|
| 1 | OK |
| 2 | OK |
| 3 | Fehler - wert ist -1 |
| 4 | OK |
| 5 | OK |

d) Der Unit-Test schlägt fehl. Passen Sie den Quellcode so an, dass der Unit-Test erfolgreich durchlaufen wird.

```
int berechneDifferenz (int zaehlerstandAlt, int zaehlerstandNeu) {

    int differenz  =  -1;

    if (zaehlerstandAlt  >=  0  &&  zaehlerstandNeu  >= 0) {

        if  (zaehlerstandAlt  <  zaehlerstandNeu)  {
            differenz  =  zaehlerstandNeu  -  zaehlerstandAlt;
        }

    }

    return differenz;
}
```

e) Im Unit-Test fehlen noch wichtige Testfälle. Geben Sie Testdaten aus zwei weiteren Äquivalenzklassen an, für welche die Methode den Rückgabewert -1 liefern sollte.

| Nr. | zaehlerstandAlt | zaehlerstandNeu | differenz |
|---|---|---|---|
| 1 | -200 | 50 | -1 |
| 2 | 50 | -200 | -1 |