Hinweis: Teilweise motiviert durch eine andere Entwicklungsumgebung sehe ich bei if-Anweisungen häufiger die folgende Formatierungsart:

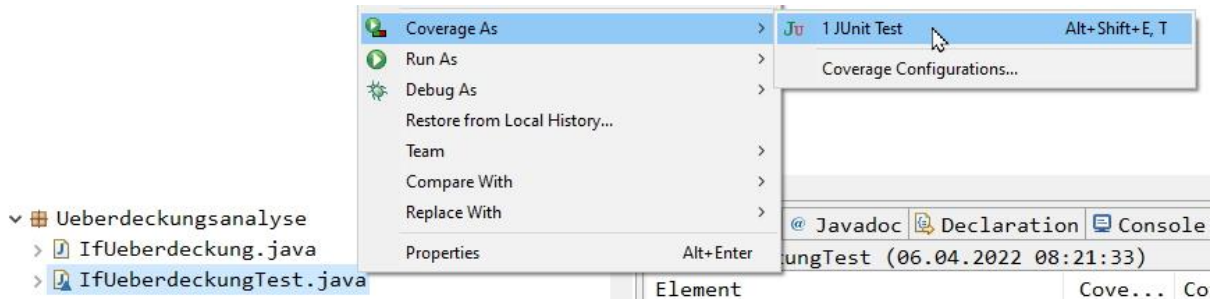
```
public class IfUeberdeckung {
  public String ueber42(int wert) {
    if(wert <= 42) return "nicht ok";
    return "ok";
  }
}
```

Diese ist aus mehreren Gründen schlecht.

Zu jeder Software werden Tests geschrieben, wie der folgende Code zeigt.

```
  @Test
  void test() {
    Assertions.assertEquals("ok",new IfUeberdeckung().ueber42(43));
  }
```

Um zu erkennen, ob es genügend Tests gibt, hilft die Messung der Testüberdeckung, die auf verschiedene Wege passieren kann (siehe [Kle19], auch Vorlesung zum Software-Engineering-Projekt). Um die von Eclipse automatisch unterstütze Messung zu nutzen, wird eine Testklasse über „Coverage As“ ausgeführt.



Die Überdeckung wird dann u. a. farblich angezeigt (grün ist genutzt, rot wurde nicht ausgeführt, gelb „so halb“). Aus der folgenden Überdeckung ist zwar ersichtlich, dass irgendwas beim if nicht ganz geklappt hat, aber u. a. die Auswirkungen sind unklar.

```
 7  public String ueber42(int wert) {
 8    String erg = "ok";
 9    if(wert <= 42) erg = "nicht ok";
10    return erg;
11  }
```

Wird der Code besser formatiert, ist das Problem des nicht ausgeführten Codes unmittelbar sichtbar.

```
 7  public String ueber42(int wert) {
 8    String erg = "ok";
 9    if(wert <= 42)
10      erg = "nicht ok";
11    return erg;
12  }
```

Leider ist die Formatierung immer noch schlecht, wenn auch für Testüberdeckungsmessungen in Ordnung. Das Problem zeigt der folgende Code, bei dem eine nachfolgende entwickelnde Person eine Warnung ergänzen wollte.

```
public String ueber42(int wert) {
  String erg = "ok";
  if(wert <= 42)
    erg = "nicht ok";
    System.err.println("zu niedriger Wert");
  return erg;
}
```

Es sollte klar sein, dass die Warnung genau beim falschen Fall ausgegeben wird. Mit geschweiften Klammern kann das Problem nicht auftreten.

```
public String ueber42(int wert) {
  String erg = "ok";
  if(wert <= 42) {
    erg = "nicht ok";
    System.err.println("zu niedriger Wert");
  }
  return erg;
}
```

Das Beispiel ist nebenbei nicht „akademisch“, es hat schon zu einem gravierenden Sicherheitsproblem geführt, wie es in https://www.spiegel.de/netzwelt/web/goto-fail-apples-furchtbarer-fehler-a-955154.html beschrieben ist. Sie schreiben Code immer so, dass er für andere entwickelnde Personen leicht les- und erweiterbar ist. Obiger Fehler passiert z. B. sehr schnell, wenn bei der Entwicklung zwischen Java und Python umgeschaltet werden muss, da Leerzeichen zur Einrückung in Python eine zentrale Bedeutung haben um die Blockzugehörigkeit definieren.

Leider gibt es an dieser Stelle oft die Frage, muss ich das auch bei der Hausarbeit berücksichtigen. Die Standardantwort ist, dass Sie die Hausarbeit so gestalten können wie Sie wollen; nur muss ich Ihnen auch keine gute Note geben.

Abschließend, ja die obige Methode könnte noch etwas kürzer programmiert werden. Das wäre auch ok. Da es aber praktisch keinen Performance-Unterschied gibt, würde das für eine Bewertung irrelevant sein.

[Kle19] S. Kleuker, Qualitätssicherung durch Softwaretests, 2. aktualisierte und erweiterte Auflage, Springer Vieweg, Wiesbaden, 2019

Kreative Wege zur Teilnahme an einer Zoom-Session eines Praktikums:

Sollten Sie an einem Online-Praktikum nicht von zuhause aus teilnehmen können oder wollen, gibt es einige kreative Wege dies durchzuführen. Die Möglichkeiten mit einem eigenen Laptop sich in eine der Lernlandschaften (SB, SL, AB, …) oder einen nicht genutzten Veranstaltungsraum zu setzen, sollten bekannt sein.

Ein weiterer Weg, den ein Student nutzt, der an seinem Hauptrechner zu Hause keine Kamera hat, der auch im Rechnerraum funktioniert ist der Ansatz, sich zweimal mit der Zoom-Session zu verbinden. Dazu wird ein zweiter Zoom-Account benötigt, den man ohne Kosten über seine private E-

Mail anlegen kann. Mit der ersten Session wird der Rechner ohne Kamera und Mikrophon in die Zoom-Session eingebunden, es ist so aber möglich Bildschirminhalte zu sehen und eigene Bildschirminhalte zu teilen. Die zweite Session findet über das Smartphone statt, wodurch es eine Kamera gibt und man so als teilnehmende Person sichtbar werden kann. Weiterhin gibt es Mikrophon und Lautsprecher, so dass die Kommunikation möglich wird. Es ist dann sinnvoll über eine passende Ablagemöglichkeit für das Smartphone nachzudenken.

Der nächste Weg besteht darin, dass eine eigene Ausstattung mit in den Rechnerraum genommen und über die USB-Ports der Hochschulrechner angebunden wird. Die Erkennung der Hardware kann durchaus etwas länger dauern, bis zu 5 Minuten, ist aber bis jetzt bei allen Experimenten (Kamera, Head-Set, Maus, Tastatur) problemlos machbar. Nach meinem Erkenntnisstand haben nicht alle Rechner, genauer Bildschirme, Lautsprecher, so dass eventuell neben einer Kamera mit Mikrophon noch ein Kopfhörer oder Headset benötigt wird. Das Kamera-Experiment ist auch in diesem Video https://youtu.be/sFzWNEYV-DQ festgehalten. Die Kamera ist etwa 12 Jahre alt: https://www.philips.de/c-p/SPZ3000_00/pc-webcam.