Übungsgruppe: Qianli Wang und Nazar Sopiha

Aufgabe 11-1:

a) 1. Sind Überlauf oder Unterlauf während der Berechnung möglich?
7. Werden alle Schleifen auf jeden Fall beendet?
8. Sind die Vergleichsoperatoren korrekt? (<, <=, >, >=)
14. Müssen Objekte mit equals() oder direkt mit == verglichen werden?
16. Ist das Indexieren von Arrays außerhalb des gültigen Bereichs möglich?

b) **9. Sind alle else-Zweige richtig behandelt? Wenn einer if-Bedingung der else-Zweig fehlt, wird der Fall richtig behandelt, wenn die if-Bedingung nicht erfüllt wird?**
Dieser Punkt kann nicht automatisiert werden, weil ob if noch ein else braucht oder ob ohne else es “richtig” behandelt wird - das ist meistens eine logische Aufgabe und logische Aufgaben können kaum automatisch geprüft werden.

c) **3. Wurden Klammern eingesetzt um Mehrdeutigkeit zu vermeiden?**
Da die Operationen unterschiedliche Prioritäten besitzen, ist es ziemlich schwer in manchen Fällen zu erkennen, ob es hier noch Klammern benötigt wird, wenn die Prioritäten von zwei Operationen ähnlich sind.

d) 1. Class and functions should be small and focus on doing one thing. No duplication of code.
2. Use a standard code formatting template.
3. Are the Null-Objects by other objects or functions invoked?
4. Omit needless and commented out code. No System.out.println statements either.
5. Werden richtige Data Types benutzt?
(**Quelle**: https://www.java-success.com/30-java-code-review-checklist-items/)
Wir haben diese Punkte gewählt, weil die nicht so schwer zu checken sind, trotzdem machen den Code viel sauberer.

e) 1. Beim Durchsichten kann man logische Paradigmen nochmal durchgehen und logische Fehler herausfinden. Tests können Logik nicht.
2. Man muss kein zusätzlichen Code entwerfen.
3. Normalerweise findet Durchsichten nicht alleine statt, dann kommuniziert das Team während des Prozesses, dadurch wird die Atmosphäre im Team verbessert werden und Wissen mitgeteilt.

Aufgabe 11-2:

b) Die Argumentation beginnt damit, dass ein Zusammenhang zwischen der Qualität von Programmierern und der Häufigkeit von goto statements in ihren Programmen erwähnt wird. Anschließend werden Alternativen erwähnt und deren Vorteile, wie cases mit indexen oder dynamische Verfahren im Allgemeinen. Den Abschluss bildet die Nennung weiterer Personen welche dem Beschriebenen zustimmen.

Die Argumentation war schlüssig, falls es möglich ist dynamische Verfahren zu einzusetzen sollte dies getan werden.

Aufgabe 11-3:

a) **Äquivalenzklassen:** F, D, C, B, A

b) (19, 21) (20, 20) (20, 40) (30, 45) (35, 55)

c) C und D: (20, 39) (20, 40)
B und C: (29, 45) (30, 45)
A und B: (30, 46) (36, 55)

D und F: (19, 21) (20, 20)
F und C: (19, 40) (20, 40)
F und B: (19, 55) (20, 55)
F und A: (19, 60) (31, 60)

d) Wir haben in dem Übergang F nach A den Rand 19 → 30 für die erste Prüfung betrachtet, für die zweite Prüfung 60 Punkten. Da wird aber B als Grade erwartet, trotzdem sind genau 90 Punkten erworben. Wenn wir aber den Rand 19 → 31 nehmen, also 91 Punkten insgesamt, läuft alles problemlos.
Hypothesen warum:
1. Wenn man 90 Punkte insgesamt erworben hat, kriegt man trotzdem B anstatt A.
Grund dafür → 8. Sind die Vergleichsoperatoren korrekt? (<, <=, >, >=)

2. Es landet in Randcase testOfBA und funktioniert nicht so richtig. Mit 91 Punkten insgesamt ist es eindeutig A.
Grund dafür → 26. Wird der Kontrollfluss richtig fortgesetzt, wenn eine Ausnahme auftritt oder abgefangen wird?