Software Engineering Homework 5

Aufgabe 1: Code Review - Verbesserungsmöglichkeiten

Bei der Überprüfung der Code-Review-Praktiken lassen sich mehrere Verbesserungsmöglichkeiten identifizieren:

1. Unklare Sprache und Ton: Die Review verwendet ungenaue und leicht herablassende Formulierungen wie "Basic Programming 101 knowledge ;)". Professionelle Code-Reviews sollten einen respektvollen, objektiven und konstruktiven Ton wahren.
2. Mangel an spezifischen Empfehlungen: Der Reviewer erwähnt potenzielle Probleme wie ineffiziente Schleifen und mögliche Sicherheitsrisiken, gibt jedoch keine konkreten Verbesserungsvorschläge oder spezifische Refactoring-Strategien.
3. Unbegründete Behauptungen: Der Reviewer deutet Leistungs- und Sicherheitsbedenken an, ohne Beweise oder spezifische Codebeispiele zu liefern, was die Glaubwürdigkeit des Feedbacks verringert.
4. Inkonsistente Formatierung: Die Review folgt keinem strukturierten Ansatz. Bewährte Praktiken sehen typischerweise eine Kategorisierung des Feedbacks vor (z.B. kritische Probleme, Verbesserungsvorschläge, positive Beobachtungen).
5. Unvollständige technische Bewertung: Die Review berührt mehrere Aspekte (Effizienz, Lesbarkeit, Sicherheit), liefert aber keine systematische, technische Aufschlüsselung der möglichen Verbesserungen.

Ein professionellerer Ansatz würde Folgendes beinhalten:

- Verwendung neutraler, spezifischer Sprache
- Bereitstellung konkreter Codebeispiele zur Verbesserung
- Erklärung der Begründung für vorgeschlagene Änderungen
- Anbieten konstruktiver, umsetzbarer Rückmeldungen
- Wahren eines respektvollen und kollaborativen Tons

Aufgabe 2: Black-Box-Testing

Für die Methode checkGroupCapacities werde ich eine umfassende Teststrategie unter Verwendung von Äquivalenzklassentests und Grenzwertanalyse entwickeln.

Spezifikationsanalyse:

- Methodensignatur: public int checkGroupCapacities(int totalStudents, int groupSize, int availableGroups)
- Rückgabe: Anzahl der zuzuweisenden Studenten
- Sonderfall: Gibt 0 zurück, wenn totalStudents <= 0
- Wirft IllegalArgumentException, wenn groupSize oder availableGroups Null oder negativ sind

Äquivalenzklassen und Grenzwerte:

1. totalStudents-Eingabe:
    - Negative Werte
    - Null
    - Positive Werte
    - Sehr große Werte
2. groupSize-Eingabe:
    - Negative Werte
    - Null
    - Positive Werte
    - Große Werte
3. availableGroups-Eingabe:
    - Negative Werte
    - Null
    - Positive Werte
    - Große Werte

Testfall-Tabelle:

Testfall-ID	Gesamtstudenten	Gruppengröße	Verfügbare Gruppen	Erwartetes Ergebnis	Begründung
TC1	-5	10	3	IllegalArgumentException	Ungültige Eingabe
TC2	0	10	3	0	Behandlung Sonderfall
TC3	15	-2	3	IllegalArgumentException	Ungültige Gruppengröße
TC4	15	0	3	IllegalArgumentException	Ungültige Gruppengröße
TC5	15	10	-1	IllegalArgumentException	Ungültige Gruppenanzahl
TC6	15	10	0	IllegalArgumentException	Ungültige Gruppenanzahl
TC7	15	5	3	0	Exakte Kapazitätsübereinstimmung
TC8	20	5	3	5	Studenten erfordern zusätzliche Gruppe
TC9	100	10	8	20	Szenario mit großer Eingabe
TC10	7	3	2	1	Ungleiche Verteilung
TC11	Integer.MAX_VALUE	10	3	Behandlung Überlauf	Extreme große Eingabe

Die Implementierung würde JUnit-5-Testmethoden erfordern, die diese Szenarien abdecken und sowohl die Berechnungslogik als auch die Ausnahmebehandlung validieren.

Aufgabe 3: White-Box-Testing in einem gesonderten Dokument