TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Software- und Multimediatechnik, Lehrstuhl für Softwaretechnologie

Wintersemester 2024/25 - Softwaretechnologie II

Kursverantwortlicher: Dr. Sebastian Götz
Übungsleiter: Dr. Dmytro Pukhkaiev

Komplex 3 – Qualitätssicherung und Testen

Ziel dieser Teilaufgabe soll es sein, einen Einblick in das Themengebiet Testen zu bekommen sowie praktische Erfahrungen mit in der Java-Community typischerweise eingesetzten Testwerkzeugen zu sammeln.

Im OPAL-Ordner zu diesem Übungskomplex finden Sie eine Java-Klasse `SimpleLinkedList`. Leider wurde diese Klasse bisher ohne Tests und sonstige Infrastruktur entwickelt, zudem sind einige der Methoden fehlerhaft.

1. Um die weitere Entwicklung gut verfolgen zu können, überführen Sie die Datei zunächst unverändert in ein dafür angelegtes Git-Repository. Legen Sie auf einer Code-Hosting-Plattform Ihrer Wahl (GitLab TU Chemnitz nutzbar mit Hochschullogin oder GitHub) ein dazugehöriges, privates Projekt an und sorgen Sie dafür, dass alle Teammitglieder entsprechende Rechte für Pull- und Push-Operationen haben.

2. Legen Sie eine `README.md` Datei im Repository an (Markdown-Format). Erläutern Sie in dieser stichpunktartig Ihre Vorgehensweise bei allen nachfolgenden Punkten.

3. Überführen Sie die bisher alleinstehende Java-Datei in ein Maven-Projekt. Achten Sie darauf, dass Sie die Konvention des Verzeichnislayouts von Maven einhalten. Erläutern Sie kurz, wie man mit Maven Kompilier- und Testvorgänge anstößt.

4. Konfigurieren Sie das Maven-Projekt so, dass Sie mit JUnit 5 Tests schreiben können sowie flüssige Assertions mit AssertJ (alternativ Google Truth) schreiben können. Erläutern Sie die Änderungen an der `pom.xml`.

5. Machen Sie sich mit dem Code-Coverage-Werkzeugen Jacoco und EclEmma vertraut und konfigurieren Sie das Plugin in Maven sowie in Ihrer IDE. Welche Arten von Abdeckung unterstützen die Werkzeuge und wie kann man sie konfigurieren?

6. Ergänzen Sie das Projekt nun um eine Testsuite für `SimpleLinkedList`. Die Testsuite sollte in der Lage sein, die noch vorhandenen Fehler in dieser Klasse zu finden. Beheben Sie diese Fehler.

7. Einige Fehler können durch dynamische Tests schwierig zu finden sein. Ergänzen Sie daher das Maven-Projekt um statische Codeanalyse: 1) aktivieren Sie zunächst alle Warnungen des Java-Compilers, 2) konfigurieren Sie im nächsten Schritt SpotBugs als Plugin. Lassen Sie die ursprüngliche, fehlerhafte Version von `SimpleLinkedList` durch dieses Setup laufen und erläutern Sie die Ergebnisse.

8. Demonstrieren Sie in einer IDE Ihrer Wahl den Debugger, indem Sie die verkettete Datenstruktur mit einem geeigneten Beispiel im `Variables` View visualisieren. Dokumentieren Sie dies als Screenshot und fügen Sie es den Erläuterungen im README hinzu.

Das README darf als die Grundlage für die Endpräsentation verwendet werden, aber kein „Copy & Paste“! Viel Spaß!