Philipps Universität Marburg

Prof. Gabriele Taentzer
Stefan John

Philipps-Universität Marburg
Fachbereich Mathematik und Informatik

Software-Praktikum
Meilenstein 1

Ausgabe: 20. April 2023
Abgabe: 04. Mai 2023

Allgemeiner Hinweis:

Achten Sie darauf, dass zur Meilensteinabnahme sämtlicher den Meilenstein betreffender Code in Ihr Repository eingecheckt und auf einem Branch gemergt ist. **Planen Sie für den Merge genug Zeit ein, damit Sie die Möglichkeit haben, auf Probleme zu reagieren.** Es muss eine lauffähige Version der Anwendung existieren, die auf einem Ihrer Rechner ausgecheckt und vorführbereit ist. Dies betrifft nicht nur der Code der Anwendung, sondern alle erstellten Artefakte wie Testsuite, Modelle und Dokumentationen.

1 Umsetzung

Entwurf und Implementierung. Beachten Sie, dass sich die Anforderungen etwas geändert haben. Eine neue Version der Spezifikation befindet sich in Ilias; Änderungen wurden markiert. Passen Sie die von Ihnen aus der Spezifikation abgeleitete Datenstruktur ggf. an die Änderungen an und setzen Sie die Datenstruktur im Code um. Erstellen Sie zudem eine lauffähige Anwendung, die eine Teilnehmerliste im vorgegebenen Format (aus einer .csv-Datei) einlesen, in Ihre interne Datenstruktur überführen und menschenlesbar auf der Konsole ausgeben kann. Eine Beispielliste finden Sie in Ilias im Ordner `Daten`. Achten Sie auf die Qualität Ihres Codes und beachten Sie die in der Spezifikation geforderten Code-Konventionen. Zur Unterstützung können Sie bspw. die in IntelliJ integrierten Code-Inspections[1] oder das Werkzeug Checkstyle[2] verwenden.

Modellieren Sie, falls noch nicht geschehen, wie die in der Spezifikation genannten Kriterien in Ihrer Anwendung repräsentiert werden sollen. Erweitern Sie Ihre Diagramme entsprechend.

Testen. Um die korrekte Umsetzung der Anforderungen zu überprüfen, müssen Sie im Rahmen des Praktikums Ihre Implementierung testen. Nutzen Sie, insbesondere später bei der Umsetzung von Anwendungsfällen, den Ansatz der testgetriebenen Entwicklung. Spezifizieren Sie als Team also welche Testfälle nötig sind, um die korrekte Umsetzung der Anforderungen zu prüfen, **bevor Sie mit der jeweiligen Implementierung beginnen**. Auf diese Weise erkennen Sie frühzeitig Uneindeutigkeiten in den Anforderungen und können diese diskutieren bzw. sich an die Person wenden, die Ihr Tutorium leitet.

[1] https://www.jetbrains.com/help/idea/code-inspection.html
[2] http://checkstyle.sourceforge.net

Führen Sie dort, wo es möglich ist, Unit-Tests durch. Führen Sie ansonsten manuelle Systemtests durch. **Für den ersten Meilenstein muss mindestens die korrekte Umsetzung des Einlesens der Teilnehmerliste durch Unit-Tests abgeprüft werden!** Dokumentieren Sie Ihre Tests wie im nächsten Kapitel beschrieben.

2 Dokumentation

Folgende Artefakte sind im Ordner `Dokumentation` Ihres Repositorys abzulegen.

Planung. Zu Beginn jedes Meilensteins ist eine Aktualisierung des in der Einführung erstellten Gannt-Diagramms gefordert. Evtl. Verzögerungen in der letzten Entwicklungsphase, Änderungen an den Anforderungen und zusätzliche Aktivitäten, die sich ggf. aus der Meilensteinabnahme ergeben, sollen hier berücksichtigt werden. Speichern Sie die jeweils aktuelle Version des Diagramms in einer neuen Datei, damit die Diagramme der einzelnen Phasen verglichen werden können.

Anwendung. Dokumentieren Sie Ihre Designentscheidungen durch geeignete Diagramme und halten Sie diese **aktuell**. Insbesondere Klassendiagramme zur Dokumentation des Domänenmodells Ihrer Anwendung sind verpflichtend. Für komplexe Abläufe können aber ggf. auch Interaktions- oder Aktivitätsdiagramme sinnvoll sein. Erstellen Sie für alle öffentlichen Klassen und Methoden Javadoc-Kommentare.

Anforderungsänderungen. Grundsätzlich sollten Anforderungen wie beschrieben umgesetzt werden. Sollten Sie aus wichtigen Gründen Anforderungen nicht wie gefordert umsetzen, dokumentieren Sie dies in einer Datei mit dem Namen `Anpassungen`.

Tests. Erstellen Sie eine Datei mit dem Namen `Testdokumentation`, in welcher Sie über das gesamte Praktikum hinweg Ihr Vorgehen beim Testen dokumentieren. Erfassen Sie in dieser für jeden Anwendungsfall, wie dieser abgetestet wurde.

Fügen Sie dazu für Unit-Tests eine Tabelle der von Ihnen getesteten Methoden ein. Geben Sie für jede Methode die Gesamtzahl der Testfälle und die Anzahl der fehlgeschlagenen Tests an und vermerken Sie, ob eine vollständige Statement-Coverage erreicht wurde (siehe Tabelle 1). Die einzelnen Testfälle müssen nicht aufgelistet werden.

Dokumentieren Sie Systemtests tabellarisch (Abb. 1). Bei Test der Benutzeroberfläche kann die Verwendung von Screenshots sinnvoll sein. In jedem Fall müssen aus der Dokumentation eines Testfalls die Vorbedingung (was war der Ausgangszustand), der Ablauf des Testfalls (welche Eingaben sind erfolgt; welche Rückmeldungen/Änderungen waren

Erfolgreiches Undo einer manuellen Änderung der Pärchenliste

Vorbedingung: Der Nutzer lässt sich eine Pärchenliste anzeigen. Die Pärchenliste enthält ein Pärchen. Die Nachrückendenliste ist leer.
Ablauf: Der Nutzer löst das vorhandene Pärchen auf. Die Teilnehmer des Pärchens werden in der Nachrückendenliste für Teilnehmer angezeigt. Der Nutzer betätigt die Undo-Funktion.
Erwartetes Verhalten: Das Pärchen in der Pärchenliste ist wieder vorhanden. Die Nachrückendenliste ist leer.
Tatsächliches Verhalten:

Abbildung 1: Beispiel für die tabellarische Dokumentation eines Testfalls.

im Verlauf zu beobachten), das erwartete Verhalten (was hätte gemäß der Anforderungen passieren müssen) und natürlich das tatsächliche Verhalten hervorgehen. Das tatsächliche Verhalten kann frei gelassen werden, wenn es dem erwarteten entspricht. Zudem sollte jeder Testfall einen sprechenden Titel besitzen, der auf einen Blick beschreibt, welche Funktionalität in ihm getestet wird.

Methode	# Tests	# Fehler	Voll. Abd.
XYZ.abc()	9	1	ja
ABC.xyz(String)	12	0	nein

Tabelle 1: Beispiel-Tabelle für Unit-Tests