Software Management Grundlagen

Prof. Dr. Manfred Brill
Prüfungsleistung

Wintersemester 2020/21

In diesem Dokument finden Sie Angaben für die Prüfung im Fach Software Management Grundlagen im Wintersemester 2020/21 und Hinweise auf Bewertungskriterien.

1 Prüfung

Termine

Für die Bearbeitung der Prüfungsleistung benötigen Sie einen `GitHub`-Account. Wir haben diese Accounts während der Präsenzphase gesammelt. Sollten Sie während der Präsenz-Woche verhindert gewesen sein berücksichtigen Sie bitte, dass wir `GitHub`-repos für die Prüfungsleistung nur bis Freitag, 18. September 2020 einrichten!

1.1 Gegenstand der Prüfung

Wir beschäftigen uns in der Prüfung mit „best practices“ in einem Software-Projekt. Für jeden Teilnehmer gibt es in der Organisation `Informatik-HS-KL` auf `GitHub` ein privates repository mit dem Namen `SMG-<<Familienname>`. Die Abgabe der Software wird in diesem `GitHub`-repo durchgeführt, weitere Einzelheiten finden Sie weiter unten.

Die in der Prüfungsleistung geforderten Builds führen Sie auf Ihrem Rechner und auf `GitHub` durch. Die von Ihnen erstellten `Jenkins`-Projekte sind Bestandteil der Abgabe. Am Ende geben Sie eine Projektdokumentation ab. Diese Abgabe erfolgt in OLAT.

1.2 Aufgaben

Schritt 1 Clonen Sie das repo `https://github.com/MBrill/ZahlenAufgabe.git`. Legen Sie ein `Eclipse`-Projekt an, so dass Sie lokal entwickeln können. Führen Sie die Anwendungsklasse `App` aus, um sicher zu stellen, dass die Klassen korrekt arbeiten. Verschaffen Sie sich einen Überblick über die Klassen.

Schritt 2 Erstellen Sie eine weitere Anwendungsklasse, in der ein Bruch auf der Konsole eingegeben wird, der gekürzt werden kann. Kürzen Sie die eingegebene Zahl und geben Sie das Ergebnis auf der Konsole aus. Achten Sie darauf, sinnvolle und aussagekräftige commit messages zu verwenden!

Schritt 3 Fügen Sie Ihre Änderungen in den `master`-branch Ihres lokalen repos ein. Erzeugen Sie ein Release „release1“ im `master`-Branch.

Schritt 4 Synchronisieren Sie ihr lokales repo (alle Branches) mit ihrem `GitHub`-repo. Verändern Sie die Datei `README.md` so, dass die von Ihnen durchgeführten Änderungen dort beschrieben werden!

Schritt 5 Als nächsten Meilenstein dokumentieren Sie die Quellen aller Klassen mit Hilfe von `Doxygen`. Erzeugen Sie damit eine `HTML`-Dokumentation.

Schritt 6 Stellen Sie sicher, dass diese Änderungen in den `master`-Branch Ihres lokalen repos eingepflegt sind. Erzeugen Sie ein Release „release2“ im `master`-Branch.

Schritt 7 Synchronisieren Sie ihr lokales repo (alle Branches) mit ihrem `GitHub`-repo. Verändern Sie die Datei `README.md` so, dass die von Ihnen durchgeführten Änderungen dort beschrieben werden!

Schritt 8 Als nächsten Meilenstein fügen Sie Log-Ausgaben in alle Klassen ein. Verwenden Sie dazu `LOGBack`. Verwenden Sie mindestens zwei verschiedene Log-Stufen. Die Konfiguration soll extern über eine Datei durchführbar sein.

Schritt 9 Fügen Sie Ihre Änderungen in den `master`-branch Ihres lokalen repos ein. Erzeugen Sie ein Release „release3“ im `master`-Branch.

Schritt 10 Synchronisieren Sie ihr lokales repo (alle Branches) mit ihrem `GitHub`-repo. Verändern Sie die Datei `README.md` so, dass die von Ihnen durchgeführten Änderungen dort beschrieben werden!

Schritt 11 Verändern Sie ihr Projekt so, dass Sie `Maven` als Build-Werkzeug einsetzen. Das Projekt muss als Ergebnis außerhalb einer IDE, in einer Konsole, gebaut werden können. Das Erzeugen der `HTML`-Dokumentation muss als Target durchgeführt werden können.

Schritt 12 Fügen Sie Ihre Änderungen in den `master`-Branch Ihres lokalen repos ein. Erzeugen Sie ein Release „release4“ im `master`-Branch.

Schritt 13 Synchronisieren Sie ihr lokales repo (alles Branches) mit ihrem `GitHub`-repo. Verändern Sie die Datei `README.md` so, dass die von Ihnen durchgeführten Änderungen dort beschrieben werden!

Schritt 14 Verwenden Sie den `master`-Branch Ihres `GitHub`-repos und erstellen Sie mit ihrer lokalen `Jenkins`-Installation ein `Jenkins`-Projekt. Das Projekt soll auf Ihrem Rechner fehlerfrei gebaut werden können, inklusive der `HTML`-Dokumentation. Im Home-Verzeichnis Ihrer Installation von `Jenkins` (Sie finden dieses Verzeichnis auch unter der Variable `JENKINS_HOME`) gibt es im Verzeichnis `jobs` ein Verzeichnis, das mit dem Namen Ihres `Jenkins`-Projekts übereinstimmt. Erstellen Sie ein Archiv dieses Verzeichnisses und Screen-Captures der Konsolen-Ausgabe von `Jenkins` eines erfolgreichen Build-Vorgangs! Dieses Archiv ist Bestandteil Ihrer Abgabe in `OLAT`!

Schritt 15 Verwenden Sie `GitHub Actions` und erstellen Sie ein Build in Ihrem `GitHub`-Repo. Verwenden Sie dabei `Maven`. Erzeugen Sie ein Release „release5“ im `master`-Branch.

1.3 Abgaben

Bestandteil der Abgabe ist Ihr `GitHub` repo mit den oben beschriebenen Branches, Releases und Commits. Änderungen auf `GitHub` die nach dem Abgabedatum erfolgen werden nicht bewertet.

Im OLAT-Kurs geben Sie zwei zip-Archive ab:

- ein Archiv mit der von Ihnen erstellten HTML-Dokumentation.
- ein Archiv Ihres Projekt-Verzeichnisses ihrer lokalen `Jenkins`-Installation wie unter Schritt 14 beschrieben.

Zusätzlich geben Sie eine `PDF`-Datei in `OLAT` ab, in der die folgenden Punkte dargestellt sind:

- Welche Veränderungen wurden an den Quellen in den einzelnen Schritten vorgenommen? Warum?
- Screen-Captures der Ausführung der zweiten Anwendungsklasse im Projekt und von Logging-Ausgaben in der Konsole oder in einer ASCII-Datei.
- Screen-Captures der Konsolenausgabe in ihrer lokalen `Jenkins`-Installation eines erfolgreichen Build-Vorgangs.
- Beschreiben Sie `GitHub Actions` und insbesondere, wie Sie mit Hilfe dieser Option ein Build auf `GitHub` durchgeführt haben!

In Tabelle 1 finden Sie eine Zusammenfassung der Branches und Tags im master-Branch ihres `GitHub`-repos.

Tabelle 1: Releases und tags im `master`-branch ihres `GitHub`-repos

Schritt	Aktivität	tag
3	Veränderung des Quelltexts	release1
6	Dokumentation	release2
9	Logging	release3
12	Build-Werkzeug	release4
15	Build mit `GitHub Actions`	release5

2 Bewertung der Prüfungsleistung

Die Bewertung der Prüfungsleistung orientiert sich an den folgende Punkten:

- Vollständigkeit der Abgabe (10%)
- Fachliche Qualität der erstellten `HTML`-Dokumentation und der Logging-Lösung (20%)
- Fachliche Qualität der Branches und Commits im `GitHub` repo (40%)
- Fachliche Qualität des Builds in ihrer lokalen `Jenkins`-Installation (15%)
- Fachliche Qualität des Builds mit `GitHub Actions` und der Dokumentation dazu (15%)

Die bei den Punkten angegebenen Anteile geben die Gewichtung in der Gesamtbewertung an.