Projekt 2

Kontext

Im letzten Semesterdrittel setzen Sie als Gruppenarbeit ein eigenes Projekt um. Die Projektidee dazu wird im "Workshop Projektidee" erarbeitet und evaluiert. Anschliessend werden in der Projektskizze die Anforderungen und Umfang des Projektes genauer spezifiziert und Investoren vorgestellt.

Einbettung in die Lernziele von PM2

Die Umsetzung des Projektes trägt zur Erreichung von folgenden Lernzielen von PM2 bei:

- Die Studierenden sind in der Lage ein Softwareprojekt mit vorgegebener Spezifikation im Team fachlich und methodisch korrekt umzusetzen, die Abläufe einzuhalten, sowie Werkzeuge konsequent einzusetzen.
- Sie arbeiten in einem Team aktiv und zielführend zusammen und übernehmen dabei Verantwortung für die Erarbeitung des gemeinsamen Projektes wie auch für den Lernfortschritt aller Teammitglieder.

Auftrag

Setzen Sie das in ihrer Projektidee beschriebene Projekt im vorgegebenen Rahmen um.

Anforderungen

- Die Anwendung muss mindestens den in der Projektskizze beschriebenen Hauptablauf plus dazu notwendige Nebenabläufe umsetzen. Die Auswahl der umzusetzenden Abläufe ist mit dem Betreuer abzusprechen.
- Es muss eine Benutzerinteraktion über eine JavaFX-Benutzerschnittstelle stattfinden. Das Model-View-Control bzw. Model-View-Presenter Pattern soll konsequent umgesetzt werden.
- Als **Architekturdokumentation** muss ein Klassendiagramm erzeugt werden. Achten Sie auf einen sinnvollen Detaillierungsgrad und eine korrekte UML-Syntax gemäss Anleitung Klassendiagramme (Link in der PM2 Kursinformation in Moodle). Erläutern Sie zudem stichhaltig und nachvollziehbar, warum Sie diese Architektur gewählt haben. Der Umfang der Erläuterung sollte zirka 0.5 - 1 A4-Seite umfassen. Die Dokumention muss Teil des Repository sein. Sie kann entweder direkt in der README-Datei enthalten sein oder muss von dieser verlinkt werden.
- Für die Klassen, welche **Domänenlogik** (Verarbeitungslogik) enthalten, müssen sinnvolle und gut dokumentierte **Tests** geschrieben werden. Die Testobjekte müssen isoliert getestet werden (Mocktesting).
- **Build-Tooling** wird korrekt verwendet. Die Anwendung muss mit dem Befehl `'gradle run'` gestartet und die Tests mit `'gradle test'` ausgeführt werden können.
- Die Projektstruktur muss sinnvoll sein und Dateien am richtigen Ort liegen.
- Das **Branching-Model** soll sinnvoll sein und konsequent angewendet werden. Die Spezifikation des Branching-Models oder ein Link darauf muss in der README-Datei enthalten sein.

- Das Projekt wird über **Issues** und **GitHub-Project** verwaltet (Mindestspalten: 'Todo' bzw. 'Backlog', 'Doing' und 'Done'; weitere wie z.B. 'Review' sind zulässig) und GitHub-Issues für die Planung des Vorgehens und Dokumentation der Änderungen und Entscheidungen verwendet. Verwenden Sie, wie in der Übung, strukturierte Beschreibungen und Labels zur Kategorisierung.
- **Pull-Requests** werden konsequent für Reviews und das Merging der Beiträge verwendet. Verlinken Sie in der README-Datei zwei Pull-Requests, in welchen sie Reviews diskutiert und Feedback integriert haben.
- Fassen sie ihren zeitlichen Aufwand wöchentlich zusammen. Wird nicht bewertet, kann aber für das Coaching nützlich sein.
- Ihr Projekt muss die im **CleanCode**-Handbuch definierten Regeln der Stufe L1, L2 und L3 erfüllen.

Umsetzung

Denken Sie bei der Umsetzung auch an folgende Punkte:

- Überprüfen Sie die grundlegenden Anforderungen aus der Projektidee.
- Einigen Sie sich in der Gruppe auf ein Klassenmodell. Dokumentieren Sie dieses.
- Definieren Sie einen Umsetzungsplan (basierend auf dem Plan in der Projektskizze).
- Erstellen Sie das Grundgerüst für Ihre Anwendung und checken den Inhalt in ein für das Projekt neu erstelltes GitHub-Repository in der Klassenorganisation ein.
- Erstellen Sie eine Gradle-Konfiguration für die Einbindung der Abhängigkeiten und das Erstellen und starten der Applikation.
- Erstellen Sie in Ihrem GitHub Repository ein GithHub-Projekt.
- Definieren und dokumentieren Sie für Ihr Projekt, welches Branching-Modell sie verwenden und nutzen Sie Pull-Requests für den Review und das Merge der Beiträge.
- Setzen Sie das Projekt inklusive Test Cases um.

Randbedingungen

- Die Java-Sprachfeatures sind auf die Version 11 zu beschränken.
- Sie können alle Grundlagen und Sprachkonstrukte verwenden, die sie im Laufe von PROG1 und PROG2 kennengelernt haben.
- Verwenden Sie weitgehend JDK-interne Komponenten. Externe Libraries sollen auf ein Minimum beschränkt werden. Einfache Libraries wie z.B. für Testing, Ein-/Ausgabe in Dateien/Console, Codierung, sowie für grafische Benutzeroberflächen (JavaFX) dürfen eingesetzt werden. Komplexe Frameworks wie z.B. Spring, Hibernate, Game-Engines, Android, etc. dürfen nur mit ausdrücklicher Erlaubnis des Betreuers / der Betreuerin eingesetzt werden. Diese wird in der Regel nur erteilt, wenn alle Teammitglieder bereits mindestens während eines Projektes mit dem Framework gearbeitet haben und somit die Einarbeitung entfällt.

Abgabe

Die Abgabe erfolgt bis zum im Wochenplan definierten Unterrichtstag um 23:59.

- Der letzte Commit im Master-Branch vor diesem Zeitpunkt wird als Abgabestand verwendet.

Bewertung

In die Bewertung fliessen die folgenden Kriterien ein:

Allgemeine Anforderungen (all-or-nothing)

Voraussetzung für Punkteerteilung: Das Programm ist lauffähig. Kriterium: Vorführung oder Test durch Betreuer. Ein nicht lauffähiges Programm erhält die Note 1.

Architektur (20%)

- Architekturdokumentation ist vorhanden, vollständig und konsistent.
- Das Klassendiagramm stimmt mit dem Code überein.
- Die Klassenstruktur ist sinnvoll, die Koppelung ist niedrig, die Aufteilung der Klassen in Module ist sinnvoll. Vererbung ist sinnvoll eingesetzt. Pattern werden eingehalten.

Software (60%)

- Das Programm besitzt die geplante bzw. mit dem Betreuer / der Betreuerin abgesprochene Funktionalität.
- Sie halten die Vorgaben hinsichtlich einsetzbarer Konstrukte und Clean Code ein und Ihr Code ist sauber dokumentiert.
- Sie haben sinnvolle Test Cases definiert (inkl. Mock-Testing) und diese können erfolgreich ausgeführt werden.

Projekt (20%)

- Ihr Vorgehen war planmässig, Fortschritte und Änderungen wurden strukturiert beschlossen und dokumentiert. Beiträge (Features, Bugs, Verbesserungen) und Entscheidungen wurden über Github-Issues verwaltet und dokumentiert.
- Beiträge von Projektmitgliedern wurden einem Review unterzogen und gezielt integriert.
- Korrekte Verwendung von Build-Tooling. Der Build funktioniert IDE unabhängig und die Anwendung kann mit gradle run gestartet bzw. mit gradle test die Tests laufen gelassen werden.
- Die Projektstruktur ist sinnvoll und Dateien liegen am richtigen Ort.
- Das Branching-Modell wurde definiert und korrekt eingehalten.
- Alle Gruppenmitglieder haben gleichermassen Code beigetragen und auf GitHub eingechecked.

