Allgemeine Informationen zur Durchführung

Im Software-Engineering Praktikum soll ein PC-Spiel basierend auf einem Brettspiel entwickelt werden. Dabei sollen die aus der Vorlesung bekannten Prinzipien zur Entwicklung von qualitativ hochwertiger Software beachtet werden.

Die Entwicklung findet in Dreier- oder Vierergruppen statt. Die Umsetzung des Projekts findet in der Zeit vom 26.04.2023 bis zum 02.08.2023 statt.

Für das Praktikum muss man sich im Uni **QIS** anmelden. Die Deadline zur Anmeldung ist der **15.05. -- 23:59 Uhr**. Diese Anmeldung ist verpflichtend. Wer sich bis zur Deadline nicht angemeldet hat, kann nicht weiter am Praktikum teilnehmen.

Umsetzung und Infrastruktur

Die Programmierung erfolgt ausschließlich in der Sprache Java, Version 11. Neben JavaFX (komplett) werden keine weiteren externen Libraries benötigt. Solltet ihr weitere externe Libraries benutzen wollen, so ist das erlaubt. Ihr seid dann jedoch dafür selbst verantwortlich, dass diese vernünftig im Build Prozess eingebunden werden, sodass es ohne manuelle Nachinstallationen auf unseren Testsystemen läuft. Ihr könnt annehmen, dass außer Java 11 und JavaFX keine weiteren Pakete installiert sind.

Die Versionsverwaltung muss mit git erfolgen.

Tutorial

Für eine Einarbeitung in die im Praktikum verwendeten Tools steht ein Tutorial zur Verfügung. Am 28.04. findet zudem ein Einführungsworkshop in die im Praktikum verwendeten Tools durch das Unternehmen Capgemini statt. Eine Teilnahme ist nicht verpflichtend, wird aber empfohlen. Details zu der Veranstaltung werden rechtzeitig im Moodle bekannt gegeben.

Infrastruktur

Für die Umsetzung des Projekts steht folgende Infrastruktur zur Verfügung:

- Gitlab zur Verwaltung des Source-Codes.

Falls ein Teilnehmer noch über keinen Account in den genannten Plattformen verfügt, kann dieser auf den jeweiligen Seiten selbstständig angelegt werden. Registrieren Sie sich dazu einen Account an unserem Gitlab: Nutzen Sie dazu Ihre offizielle E-Mail-Adresse an der Uni-Lübeck als E-Mail-Adresse und den lokalen Teil dieser E-Mail-Adresse als Benutzername. Wenn Sie also die E-Mail-Adresse `erika.mustermann@student.uni-luebeck.de` haben, registrieren Sie sich mit dem Benutzernamen `erika.mustermann`. Teilnehmer der Vorlesung im letzten Semester haben bereits einen Account in diesem Gitlab und können diesen weiter nutzen.

Entwicklungsumgebung

Für die Entwicklung ist keine spezifische Entwicklungsumgebung festgelegt. Wir empfehlen die Verwendung einer bekannten IDE wie Eclipse, IntelliJ IDEA oder Netbeans, da für diese IDEs in der Regel viele Erweiterungen und Dokumentation verfügbar sind. Das Build-Management muss in jedem Fall via Maven erfolgen.

Tutoren und Präsenzzeiten

Die Tutoren stehen zur persönlichen Betreuung Mittwochs von 12 bis 18 für euch bereit. Auch allgemeine Fragen, insbesondere zu organisatorischen Themen können dort besprochen werden.

Ausscheiden aus dem Praktikum

Sollte ein Gruppenmitglied auf eigenen Wunsch aus dem Praktikum ausscheiden werden je nach Ausstiegszeit die Anforderungen an die Gruppe individuell angepasst. In diesem Fall nehmen die Gruppenmitglieder bitte unmittelbar Kontakt mit Ludwig Pechmann und Philipp Bende auf. Ebenso ist diesen das Ausscheiden durch das Gruppenmitglied unverzüglich mitzuteilen.

Sollte ein Gruppenmitglied offenkundig nicht willens sein sich in das Praktikum einzubringen, kann das Mitglied aus dem Praktikum ausgeschlossen werden. In diesem Fall können sich die weiteren Gruppenmitglieder an Ludwig Pechmann und Philipp Bende wenden. Auch in diesem Fall können die Anforderungen individuell für die verbleibende Gruppe angepasst werden.

Ein Ausschluss oder Ausscheiden aus dem Praktikum wird als Nicht-Bestehen gewertet und entsprechend an das Prüfungsamt gemeldet.

Abgabe

Im Praktikum finden 3 Releases statt, bei denen jeweils der bisherige Stand des Projekts im Gitlab "abgegeben" und bewertet wird.

Zwischenabgaben

Release I

Deadline 28.05. -- 23:59

- Das Team ist im QIS angemeldet
- Es existiert ein Projekt im Gitlab (Fork vom Template)
- Alle Teammitglieder sind im Gitlab in eurem Projekt Mitglieder
- Die Anforderungen sind aufgenommen und dokumentiert (Story Cards, etc.)
- Es gibt einen Projektplan (Arbeitspakete, Zeitplanung, Gantt-Chart, Meilensteine, etc.)
- Es gibt eine Verantwortlichkeitsverteilung im Team und diese ist schriftlich festgehalten. Es könnte z.B. ein Teammitglied für Architektur, ein weiteres für Testing und ein drittes für die Projektleitung verantwortlich sein. Diese Verantwortlichkeiten bedeuten allerdings nicht, dass man seine

Verantwortlichkeit alleine erledigen muss, sondern eher, dass man darauf zu achten hat, dass die jeweilige Aufgabe vom Team allgemein erfüllt wird.

Release II

Deadline 25.06. -- 23:59

- Es existiert ein erkennbares minimales Spiel
- Das Spiel kann von 2 menschlichen Spielern gespielt werden
- Das Spiel ist "per Hand" spielbar, d.H. Regeleinhaltung wird noch nicht unbedingt vom Computer überprüft
- Es existiert eine GUI und die Steuerung des Spiels mit der Maus funktioniert
- Es gibt einen "Open Issues Report" in dem noch zu erledigende Arbeiten, Bugs, etc. dargestellt sind.

Die oben genannten Zwischenabgaben fließen nicht in die Bewertung des Praktikums ein. Bei grober Nichterfüllung der Anforderungen wird jedoch eine Nachbesserung innerhalb der nächsten 10 Tage gefordert. Erfolgt diese nicht oder nicht hinreichend, scheidet die Gruppe vorzeitig aus dem Praktikum aus.

Wichtig: Die Zwischenabgaben dienen dazu einen Minimalfortschritt im Projekt sicherzustellen und euch auf mögliche Schwachstellen in eurem Projekt hinzuweisen. Ein Erreichen der Endabgabe garantiert weder ein Bestehen des Praktikums noch eine gute Note.

Endabgabe

Release III

Deadline 02.08. -- 12:00

- Das vollständige Spiel inklusiver aller Pflichtfeatures, ausreichend Wahlfeatures und Zusatzfeatures ist implementiert, getestet und spielbar.
- Das gesamte Projekt muss inklusive aller Doku/Artefakte/Anleitungen/etc. im Gitlab abgegeben werden.
- 20 Minütige Abschlusspräsentation

Abschlusspräsentation

In der Abschlusspräsentation soll das Produkt vorgestellt werden. Ihr habt dabei die Möglichkeit die Funktionalität zu demonstrieren und auf Besonderheiten eures Produkts hinzuweisen.

Abgabeformat

An den Schlusstagen der Releases, inklusive Endabgabe, muss der zu überprüfende/bewertende Stand mittels Tags (`r1`,`r2`,...) im git-Repository gekennzeichnet werden. Zusätzliche Dokumente (insbesondere die vollständige Dokumentation) sind als PDF-Dokumente `Bedienungsanleitung.pdf`, `Anforderungsanalyse.pdf` und `Architektur.pdf` im Hauptordner des git-Repositories zur Verfügung zu stellen.

Details zur Umsetzung der Anforderungen

Für das zu implementierende Spiel gelten die Regeln in der pdf-Datei "Regelwerk" im Moodle. Im Praktikum müssen eine Reihe von Features umgesetzt werden. Diese gliedern sich in Pflicht- und Wahl und Zusatzfeatures. Die genaue Beschreibung findet sich unter Features

Pflichtfeatures müssen in jedem Fall komplett umgesetzt werden.

Wahlfeatures sind jeweils mit einer Punktezahl versehen. Es müssen Wahlfeatures im Wert von 30 Punkten umgesetzt werden. Umsetzen von mehr als 30 Punkten kann zu Bonuspunkten führen. Die korrekte Umsetzung der Features hat jedoch einen höheren Stellenwert als die Anzahl umgesetzter Features.

Zusatzfeatures werden im Laufe des Praktikums (nach Release II) bekannt gegeben. Von diesen müssen von 3er-Gruppen eins und von 4er-Gruppen zwei umgesetzt werden. Alle weiteren Zusatzfeatures gelten als Wahlfeatures und sind 5 Punkte wert.

Bewertungskriterien

Folgende Kriterien fließen in die Bewertung ein. Die Endnote bestimmt sich durch die Gesamtzahl der Punkte, die erreicht wurde. Für ein Bestehen ist es aber zudem Voraussetzung, dass in den Kategorien 1.,2.,3. jeweils mindestens die Hälfte der Maximalpunktzahl erreicht wurde.

1. **Produktbezogene Qualitätsaspekte (50 P.)**
    1. Funktionalität und Zuverlässigkeit (30 P.)
        1. Implementierung aller Pflichtfeatures (14 P.)
        2. Implementierung ausgewählter Wahlfeatures (6 P.)
        3. Fehlerfreiheit der implementierten Funktionen (10 P.)
    2. Gebrauchstauglichkeit (15 P.)
        1. Verständlichkeit, Erlernbarkeit, Bedienbarkeit und Attraktivität des Produkts
        2. Abfangen von Bedienfehlern
    3. Effizienz/Performanz (5 P.)
        1. Keine unverhältnismäßig lange Rechen-/Antwortzeit
        2. Angemessener Ressourcenverbrauch (Arbeitsspeicher etc.)
2. **Entwicklungsbezogene Qualitätsaspekte (30 P.)**
    1. Stabile und modifizierbare Architektur (10 P.)
    2. Testing (8 P.)
        1. Testqualität
        2. Prozentuale Testabdeckung
    3. Einhaltung Metriken (4 P.)
    4. Codingstyle (4 P.)
        1. Einhaltung Codierungskonventionen
        2. Verständlicher Code (sprechende Variablen-/Methodennamen, Kommentare wo nötig)
        3. Einfacher Code (wo möglich)
    5. Versionsverwaltung (4 P.)
        1. Sprechende Commit-Nachrichten
        2. Sinnvolle Commitgrößen
3. **Dokumentation (20 P.)**
    1. Lückenlose Dokumentation der Anforderungen (Usecase-Diagramm, Storycards) (3 P.)
    2. Qualitativ hochwertige Dokumentation der Architektur inkl. Begründung (7 P.)

3. Bedienungsanleitung für das Produkt (2 P.)
4. Javadoc-Dokumentation (4 P.)
5. Abschlusspräsentation (4 P.)

Nicht-funktionale Anforderungen (jedes Release)

Codequalität

- Einhaltung der Metriken wie im Abschnitt *Codingstyle, Metriken, Testabdeckung* beschrieben
- Einhaltung Style-Convention wie im Abschnitt *Codingstyle, Metriken, Testabdeckung* beschrieben
- Kommentierung des Codes, wo notwendig

Testabdeckung

- Abdeckung von 80 % des Codes (ohne GUI Klassen) durch JUnit-Tests

Dokumentation

- Stets aktualisierte Anforderungsdokumentation via Story-Cards, Anwendungsfalldiagrammen
- Stets aktualisierte Dokumentation der Architektur unter Zuhilfenahme von UML-Diagrammen (Klassen-, Objekt-, Sequenz-, Zustand-), wo sinnvoll.
- Stets aktualisierte Javadoc-Dokumentation des Programms
- Stets aktualisierte Dokumentation der Programmverwendung (Bedienungsanleitung)

Beteiligung in Moodleoverflows

Die Stackoverflow-artigen Moodle-Foren (Moodleoverflows) dienen der gegenseitigen Hilfe der Studierenden und sorgen für effektivere Softwareentwicklung über alle Gruppen hinweg. Durch Beteiligung an den Moodleoverflows können bis zu 5 Bonuspunkte erreicht werden. Diese Punkte werden am Ende des Praktikums anhand der Aktivität der Nutzer in den Moodleoverflows vergeben.

Dokumentation

Die Doku ist kein Wahlfeature!

Anforderungsdokumentation

Die Anforderungen des Produkts sollen übersichtlich in einem Anwendungsfalldiagramm dargestellt werden. Die einzelnen Anforderungen sollen detailliert auf Story-Cards festgehalten werden, auf die während des Praktikums zurückgegriffen werden soll.

Bedienungsanleitung

In diesem Dokument sollen alle Funktionen des Systems kurz vorgestellt werden und erklärt sein, wie diese angesteuert werden können.

Generell gilt: Sind Features nicht korrekt in der Bedienungsanleitung beschrieben und werden Sie bei der Bewertung nicht entdeckt, werden sie als nicht vorhanden bewertet. Analoges gilt, wenn sich die Bedienung

der Features nicht unmissverständlich aus der Bedienungsanleitung ergibt.

Architekturdokumentation

In diesem Dokument sollen die grundsätzlichen Ideen hinter der gewählten Architektur dargelegt werden. Es sollte erklärt werden, in welche Komponenten das System eingeteilt wurde, welche Klassen zur Umsetzung der einzelnen Komponenten dienen und welche Kommunikation zwischen den einzelnen Klassen stattfindet. Außerdem sollte klar werden, in wie fern die gewählte Architektur eine Erweiterbarkeit und einfache Wartbarkeit des Programms gewährleistet.

Für die Dokumentation kann die Verwendung von UML-Diagrammen sinnvoll sein.

Javadoc-Dokumentation

Alle öffentlichen Klassen, Methoden (abgesehen von Gettern und Settern) und Attribute sollten Javadoc-konform dokumentiert sein.

Codingstyle, Metriken, Testabdeckung

Das Template verwendet die Plugins PMD und JaCoCo zur Generierung von Reports über Metriken, Codestyle und Testabdeckung. Die Verwendung der Plugins und der resultierenden Reports wird im Tutorial demonstriert.

Es wird eine Testabdeckung des gesamten Codes von 90% Instruction Coverage erwartet. (Dabei dürfen GUI-Klassen ausgeschlossen werden. Weitere Details dazu finden sich im Tutorial.)

Im Template wird unser PMD-Regelsatz eingebunden. Diese Regeln sind für den Code und die Testfälle einzuhalten.