Übungen zu „Einführung in Software Engineering“ (WS 15/16)
Prof. Dr. Barbara Paech, Marcus Seiler
http://se.ifi.uni-heidelberg.de

Übungsblatt 2 (20.10.2015)

Pair-Programming, Java, Statische Codeprüfung, Metriken

In dieser Übung:

- ✓ Wenden Sie die Prinzipien des Pair-Programmings an.
- ✓ Vertiefen Sie Ihre Kenntnisse über die Programmiersprache Java.
- ✓ Lernen Sie die Java-Programmierrichtlinien kennen.
- ✓ Analysieren Sie Ihren programmierten Java Code.
- ✓ Berechnen Sie Metriken für Ihren programmierten Java Code.

Abgabe Ihrer Lösung in Moodle
Um uns in die Lage zu versetzen, Ihre Lösungen schnell zu korrigieren, bitten wir Sie, für alle zukünftigen Übungen folgende Konventionen für Ihre Abgabe einzuhalten:
Benennen Sie Ihre Lösung stets mit nachname-aufgabennummer, beispielsweise **mueller-a12**.
Diese Namenskonvention gilt für die Abgabe von Eclipse-Projekten und für die Abgabe von PDFs.
Sollten Sie in einer Aufgabe sowohl ein Eclipse-Projekt als auch ein PDF abgeben müssen, dann speichern Sie bitte beides (Eclipse-Projekt und PDF) in einem zip-Archiv.

Versionsverwaltung mit Git:
Zum Verwalten von unterschiedlichen Versionen einer Software wird im Laufe des Semesters das Versionsverwaltungswerkzeug Git eingesetzt. Eine Anleitung für die Konfiguration und Nutzung von Git finden Sie unter „Tutorials“ in Moodle.

Aufgabe 2.1: Pair Programming

Präsenz: Ja	**Punkte:** 8	**Team: Ja (2)**	**Projekt:** Nein	**Testat**

1. Erweitern Sie Ihre Anwendung **„MyFirstMovieManager“** aus Blatt 1 durch **Pair-Programming** (zwei Personen programmieren gleichzeitig an einem Computer). Protokollieren Sie mit, wann Sie die Tastatur gewechselt haben und wer von Ihnen was getan hat. Der *Observer* fasst kurz zusammen, was der *Driver* in seiner Zeit gemacht hat.
2. Implementieren Sie eine Funktionalität zum Ausleihen von Filmen wie folgt:
   a. Legen Sie eine **neue Klasse** `Customer` an, welche Informationen zu einem Kunden speichert. Ein `Customer` soll einen Namen (Typ String) und eine Liste ausgeliehener Filme haben.
   b. Die Klasse `Movie` soll ein **zusätzliches Attribut** `customer` (Typ `Customer`) bekommen, welches speichert, von welchem Kunden der Film ausgeliehen wurde.
   c. Erweitern Sie die Klasse `Customer` um eine Operation `loanMovie()`, welche als Übergabeparameter einen auszuleihenden Film erhält. In der Implementierung der Operation muss sichergestellt werden, dass der Film nicht schon von einem anderen

Kunden ausgeliehen ist. Nur dann kann der Film auch tatsächlich vom Kunden ausgeliehen werden, sonst soll eine entsprechende Fehlermeldung erzeugt werden.

   d. Erweitern Sie die Operation `loanMovie()` so, dass ein Kunde **nur 5 Filme gleichzeitig** ausleihen darf. Implementieren Sie eine entsprechende Meldung, die warnt, wenn versucht wird mehr als 5 Filme auszuleihen.

3. Schreiben Sie ein Hauptprogramm mit einer `main`-Operation, in welchem Sie die neue Funktionalität ausführlich testen. Hierbei müssen die beiden Fehlermeldungen („Film ist schon an Kundin <Name der Kundin> verliehen“ und „Kunde <Name des Kunden> hat schon 5 Filme ausgeliehen“) mindestens einmal erzeugt werden.
4. Beschreiben Sie Ihre Erfahrungen, die Sie während des Pair-Programming gemacht haben.
5. Speichern Sie das Projekt in Ihrem Git-Repository (mit der „Share Project“ Funktion und anschließendem „Add to Index“ und „Commit“)!

Ergebnis:

Speichern Sie das Ergebnis als.zip-Datei bis **Montag 26.10.2015 um 10.00 Uhr** in Moodle. Bestehend aus:

- Exportiertes Eclipse-Projekt mit `Customer` Implementierung
- PDF mit der Beschreibung des Ablaufes des Pair-Programming (Protokoll) einschließlich des Erfahrungsberichts

Es ist ausreichend, wenn eines der beiden Teammitglieder die Ergebnisse in Moodle abgibt.

Aufgabe 2.2: Java – Anwendung (2)

Präsenz: Nein	**Punkte:** 5	**Team:** Nein	**Projekt:** Nein	**Testat**

Ihre Anwendung **„MyFirstMovieManager“** aus Aufgabe 2.1 soll zusätzlich das Anlegen eines spezialisierten `Performers`, nämlich eines `Directors`, anbieten. Ein `Director` kann sowohl in einem Film mitspielen als auch die Regie führen.

1. Der `Director` soll als neue Klasse realisiert werden. Der `Director` erbt alle Attribute, Operationen und Assoziationen vom normalen `Performer`, unterscheidet sich aber durch das neue Attribut der `Anzahl der Movies`, bei der er Regie geführt hat.
2. Bei der Erstellung eines `Movies` sollte man gleich einen `Director` zuweisen können.
3. Speichern Sie die Änderungen das Projekt in Ihrem Git-Repository („Add to Index“ und „Commit“)!

Ergebnis:

Bitte speichern Sie Ihr exportiertes Eclipse-Projekt in gepackter Form als .zip-Datei bis **Montag 26.10.2015 um 10.00 Uhr** in Moodle.

Checkstyle:
Das Eclipse Plug-In „Checkstyle“ hilft bei der Programmierung von Java Code, der die Java Code Conventions einhält. Eine Anleitung für dieses Plug-In finden Sie unter „Tutorials“ in Moodle.

Google CodePro AnalytiX:
Das Eclipse Plug-In „Google CodePro AnalytiX“ analysiert Java Code und hilft potentielle Probleme zu finden, z.B. mögliche Fehler oder „Bad Smells“. Eine Anleitung für dieses Plug-In finden Sie unter „Tutorials“ in Moodle.

Aufgabe 2.3: Statische Codeprüfung

Präsenz: Nein	**Punkte:** 8	**Team:** Nein	**Projekt:** Nein	**Testat**

1. Machen Sie sich kurz mit den „Java Code Conventions“ vertraut, die Sie in Moodle finden. Eine Übersicht über die wichtigsten Java Code Conventions mit Beispielen finden Sie hier:

   http://www.cs.umbc.edu/courses/undergraduate/202/spring11/projects/coding-standards.shtml
2. Benutzen Sie das Eclipse Plug-In „Checkstyle“ und wenden Sie es auf Ihren Quellcode aus Aufgabe 2.2 „Java – Anwendung (2)“ an, d.h. formatieren Sie den Quellcode so, dass er den Konventionen entspricht.

3. Benutzen Sie die Funktion „Audit Code Using ...“ mit dem Regelsatz „Potential Errors and Refactorings“ des Eclipse Plug-Ins „Google CodePro AnalytiX“ und wenden Sie sie auf Ihren Quellcode aus Aufgabe 2.2 „Java – Anwendung (2)“ an, d.h. ändern Sie den Quellcode so, dass er keine Audit-Regeln verletzt.

Ergebnis:
Bitte speichern Sie den formatierten Quellcode als vollständig exportiertes Eclipse-Projekt als .zip-Datei bis **Montag 26.10.2015 um 10.00 Uhr** in Moodle.

Aufgabe 2.4: Metriken

Präsenz: Nein	**Punkte:** 4	**Team:** Nein	**Projekt:** Nein	

1. Benutzen Sie die Funktion „Compute Metrics“ des Eclipse-Plug-Ins „Google CodePro AnalytiX“ und erheben Sie folgenden Metriken für Ihre Ergebnisse aus Aufgabe 2.2 „Java – Anwendung (2)“:
   - Anzahl der Quelltextzeilen in allen Operationen einer Klasse
   - Durchschnittliche Anzahl Quelltextzeilen pro Operation einer Klasse
   - Durchschnittliche Anzahl Parameter einer Klasse
   - Anzahl Attribute in einer Klasse
   - Anzahl Operationen in einer Klasse

   Dokumentieren Sie die Metriken in einer Textdatei. Diese Textdatei soll **für jede von Ihnen neu erstellte Klasse** die oben genannten Metriken beinhalten. **Vergleichen** Sie anschließend die Metriken Ihrer erstellten Klassen miteinander, d.h. dokumentieren Sie die Gemeinsamkeiten und erklären Sie die Unterschiede in der gleichen Textdatei.

2. Was bewirkt Ihrer Meinung nach die Erhebung von Metriken von Quellcode? Sammeln Sie einige Vor- und Nachteile, die die Erhebung von Metriken hervorbringt.

Ergebnis:
Speichern Sie bitte eine PDF-Datei mit den erhobenen Metriken und den Vor- und Nachteilen bis **Montag 26.10.2015 um 10.00 Uhr** in Moodle.

Aufgabe 2.5: Vorbereitung Test

Präsenz: Nein	**Punkte:** 15	**Team:** Nein	**Projekt:** Nein	

Bereiten Sie sich auf die Inhalte der nächsten Vorlesung vor, indem Sie die Abschnitte **19.1 „Begriffe und Grundlagen des Tests“** bis einschließlich **19.4 „Die Auswahl der Testfälle“** aus dem Kapitel **19 „Programmtest“** im Software Engineering Buch von Ludewig und Lichter lesen (siehe PDF in Moodle). Schauen Sie insbesondere den Abschnitt zu Äquivalenzklassentests an, um Sie dann in der Vorlesung anwenden zu können.

Tragen Sie anschließend die Inhalte zu den folgenden 3 Punkten in das Trello Board „Vorbereitung Test“ des Teams ISW ein:

- Wichtige Begriffe (und ihre Definitionen) im Umfeld des Testens, insbesondere die unterschiedlichen Testarten (Klassifikationen)
- Techniken zur Testfallauswahl
- Vor- und Nachteile des Testens

Für jeden der 3 Punkte ist eine eigene Liste vorhanden, in die Sie die Inhalte eintragen können.
Diese Aufgabe ist also wieder eine Crowd-Sourcing-Aufgabe, d.h. Sie erstellen mit allen VorlesungsteilnehmerInnen gemeinsam diese Übersicht und können vorhandene Einträge bearbeiten und verbessern. Sie lernen am meisten, wenn Sie sich aktiv beteiligen. Die Inhalte des Boards „Test“ werden in der Vorlesung besprochen (wenn nichts drin ist, wird auch nichts besprochen).

Ergebnis:
Tragen Sie bitte Ihre Inhalte bis **Montag 26.10.2015 um 10.00 Uhr** in das Trello Board ein.