Proseminar Programmiermethodik

Universität Innsbruck - Institut für Informatik

A. Blaas, M. Mayerl, B. Murauer, M. Tschuggnall

Blatt 8: Debugging und Refactoring

Diskussionsteil

a) ★ In diesem Diskussionsteil lernen Sie einige Hilfsmittel kennen, die Eclipse in puncto Debugging und Refactoring zur Verfügung stellt. Importieren Sie zunächst das Projekt `Sheet08Discussion.zip` (OLAT) in Ihre Entwicklungsumgebung und erledigen Sie die nachfolgenden Aufgaben.

b) ★ Die Klasse `EmployeeProcessor.java` (OLAT) liest zufällig generierte Angestellte eines sehr großen Unternehmens ein. Unter den 10.000 Angestellten befindet sich eine Person, die zuviel verdient (mehr als 3.000). Finden Sie mithilfe des Debuggers heraus, um welche es sich dabei handelt. Der Source-code dieser Aufgabe darf dabei nicht verändert werden.

> **Hinweis** ⚠
>
> Durch das Setzen von `Breakpoints` an der richtigen Stelle lässt sich diese Aufgabe recht schnell lösen. In `Eclipse` gibt es die Möglichkeit, `conditional breakpoints` zu setzen. Informieren Sie sich im Internet über deren Anwendung und machen Sie davon Gebrauch, um nicht alle 10.000 Angestellten manuell überprüfen zu müssen.

c) ★ Die Klasse `SalaryCalculator.java` (OLAT) berechnet das Gesamtgehalt der übergebenen Angestellten. Die Funktionalität zur Gehaltsberechnung befindet sich hier allerdings innerhalb der `main`-Methode. Ihre Aufgabe ist es, diese Funktionalität in eine Methode auszulagern.

> **Hinweis** ⚠
>
> So banal die Aufgabe auch ist, lehnen Sie sich zurück und lassen Eclipse den Rest erledigen. Markieren Sie hierfür die Zeilen 17-20 (inklusive) und drücken Sie die Tastenkombination Shift + ALT + M zum Extrahieren der Methode.

d) ★ In der Klasse `ArithmeticOperations.java` (OLAT) sind sehr banale Methoden zur Realisierung von arithmetischen Operationen vorhanden. Zu den besagten Methoden existieren bereits Unit-Tests (`ArithmeticOpsTest.java` (OLAT)). Die Namen der Methoden sind allerdings völliger Quatsch und sollten dringend ausgebessert werden.

> **Hinweis** ⚠
>
> Verwenden Sie auch hier wieder ein nettes Eclipse-Feature, mit welchem Sie die Methodennamen an einer Stelle ändern können, ohne dass sie alle weiteren Java-Dateien durchstöbern müssen, um dort den Namen ebenfalls auszutauschen. Markieren Sie hierfür den Methodennamen, drücken Sie die Tastenkombination Shift + ALT + R und geben dann einen neuen Namen für die Methode ein.

e) ★★ Lesen Sie sich zunächst den Foliensatz zum Thema `Refactoring` durch. Untersuchen Sie nun Ihre Lösungen zu den vorhergehenden Übungszettel und identifizieren Sie `bad smells` in Ihrem Code[1]. Picken Sie sich nun mindestens vier solcher `code smells` heraus, beschreiben Sie dann, worin der `code smell` besteht und refactoren Sie schlussendlich die besagten Stellen im Sourcecode. Beschreiben Sie dann außerdem, warum Ihre Refactoring-Maßnahmen sinnvoll bzw. nützlich sind.

Hinweis ⚠

Ergänzen Sie Ihr Wissen aus den Vorlesungsfolien mit diesen zusätzlichen Quellen:

- https://www.refactoring.com/
- https://sourcemaking.com/refactoring

Übungsteil (selbstständig zu lösen)

Aufgabe 1 (Debug or not debug) [2 Punkte]

"Debugging is like being the detective in a crime movie where you are also the murderer." (Filipe Fortes)

Importieren Sie das Projekt `ex1.zip` (OLAT) in Ihre Entwicklungsumgebung. Bei der Implementierung wurden alle möglichen Konventionen verletzt sowie äußerst unvorteilhafte Bezeichner gewählt. Ihre Aufgabe besteht nun im Folgenden:

a) *0.5 Punkte* Finden Sie raus, was dieser Code macht bzw. machen soll.

Hinweis ⚠

Der Debugger könnte an dieser Stelle hilfreich sein.

b) *1.5 Punkte* Nutzen Sie die vorgestellten Refactoring-Mechanismen von `Eclipse`[2] und 'entwirren' Sie diesen Code.

Aufgabe 2 (The art of refactoring) [6 Punkte]

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." (Martin Golding)

Eine Softwarefirma benötigt ein Programm, welches die Gehälter ihrer Angestellten berechnet. Es wird dabei zwischen mehreren "Berufsgruppen" unterschieden, deren Gehälter sich aus unterschiedlichen Faktoren zusammensetzen. Wie die Berechnung der unterschiedlichen Gruppen aussieht, wird im Folgenden beschrieben:

[1] Sie finden keine? Sehr, sehr, sehr unwahrscheinlich :D
[2] Bzw. der IDE Ihres Vertrauens

- Alle Berufsgruppen erhalten ein monatliches Grundgehalt von 1.200 EUR.
- Hausmeister erhalten eine zusätzliche Gefahrenzulage von 300 EUR monatlich.
- Softwareentwickler mit abgeschlossenem Bachelorstudium erhalten 600 EUR zusätzlich im Monat. Ein abgeschlossenes Masterstudium bedeutet 900 EUR mehr im Monat. Ein Entwickler ohne Universitätsabschluss, aber dafür mit mindestens 10-jähriger Berufserfahrung, verdient 750 EUR mehr im Monat. Universitätsabschlüsse und Berufserfahrung können nicht kombiniert werden: D.h., es zählt jeweils der größere Bonus.
- Projektmanager erhalten das doppelte Grundgehalt[3] sowie 50 EUR Vertrauensbonus mal Anzahl der Jahre, die sie schon in der Firma sind (Ein Manager, der also schon 10 Jahre für die Firma tätig ist, erhält also 500 EUR zusätzlich).
- Ein Praktikant wird nur dann bezahlt, wenn er gute Arbeit geleistet hat. Nämlich mit einem Taschengeld von 400 EUR.

Importieren Sie das Projekt in Ihre Entwicklungsumgebung, welches die beschriebene Gehälterberechnung umsetzt. Die Implementierung zählt leider nicht zu den Sternstunden des Entwicklers.

a) *2 Punkte* Refactoren Sie den Code der `Employee`-, bzw. `Trainee`-Klasse.

b) *2 Punkte* Refactoren Sie auch den Code der `EmployeeManager`-Klasse .

Beachten Sie dabei folgendes:

- Führen Sie eine geeignete Vererbungshierarchie ein.
- Vermeiden Sie redundanten Code.
- Halten Sie sich an Namenskonventionen.
- Gestalten Sie Ihre Implementierung so erweiterbar wie möglich.
- Vermeiden sie das ”Hartcodieren” von Werten.

c) *2 Punkte* Testen Sie Ihre Implementierung mit `Unit-Tests`. Halten Sie sich dabei an das F.I.R.S.T-Prinzip, welches Sie aus der Vorlesung kennen sollten!

Aufgabe 3 (Bugs, bugs, bugs) [2 Punkte]

”If debugging is the process of removing bugs, then programming must be the process of putting them in.” (Edsger Dijkstra)

Falls Sie die vorherige Aufgabe ausgelassen haben sollten, dann importieren Sie bitte jetzt das Projekt `ex2_3.zip` (OLAT) in Ihre Entwicklungsumgebung. In diesem Projekt finden Sie das `package com.self.ex3.errors` welche Klassen mit einigen (Programmier)Fehlern enthalten, die im Laufe dieses Semesters von den Kursteilnehmern gemacht wurden. Zeigen Sie, dass Sie aus Ihren Fehlern (bzw. den Fehlern anderer) gelernt haben[4]:

a) *0.5 Punkte* Identifizieren Sie die Fehler und beschreiben Sie im Original-File (Kommentare), worin die Fehler bestehen und welche Auswirkungen sie hätten.

[3] Warum auch immer

[4] Falls Sie jetzt denken, Sie haben keine Fehler gemacht...Hmmm...Naja...dann belügen Sie sich höchstwahrscheinlich selbst ;)

b) *1.5 Punkte* Kopieren Sie nun alle Klassen im package `com.self.ex3.errors` in ein neues package `com.self.ex3.errors.corrected` und bessern Sie die Fehler aus.

Hinweis ⚠

Die Klassen im besagten package machen insgesamt keinen Sinn. Es muss am Ende kein sinnvolles Programm rauskommen, welches irgendetwas Sinnvolles berechnet. Die Aufgabe gilt als erledigt, wenn Sie die Fehler gefunden, beschrieben und behoben haben.

Aufgabe 4 (The final chapter) [0 Punkte]

Das Ende naht. Da es sich bei diesem Proseminar um das letzte vor dem großen Finale (der Klausur) handelt, haben Sie jetzt noch einmal die Gelegenheit, Ihren Proseminarleiter mit Fragen zu löchern. Gehen Sie also nochmals in sich und notieren Sie sich dann die Punkte, über die Sie noch gerne etwas Klarheit hätten. Tragen Sie dann Ihre Fragen in ARS-Nova[5] ein. Wir werden sie dann anstelle des Diskussionsteils gemeinsam durchgehen. Nutzen Sie die Gelegenheit, Fragen zu stellen. Denn danach stellen WIR die Fragen ;)

Dies war der letzte Übungszettel. Hoffentlich haben Sie einiges mitgenommen aus diesem Kurs. Falls Sie noch Fragen, Anregungen, Kritik, Verbesserungsvorschläge oder gar Lob anbringen möchten, dann tun Sie das bitte :)

Wir ersuchen Sie außerdem, uns Ihre Meinung zum Kurs mitzuteilen, indem Sie den folgenden Fragebogen zur Vorlesung und dem Proseminar ausfüllen: https://docs.google.com/forms/d/e/1FAIpQLScYqvWKBxrfoJBBwrlkAs3ROcmM95x6CFCGP9tLevfT50k8tA/viewform?usp=sf_link

Sie würden uns damit einen großen Gefallen erweisen und könnten damit aktiv die Qualität des Kurses weiter steigern. Die Studenten nach Ihnen werden es Ihnen danken :)

Ansonsten wünschen wir euch noch viel Erfolg bei den anstehenden Klausuren (ganz speziell für Programmiermethodik xD) und im weiteren Studienverlauf.

Euer Programmiermethodik-Team.

Wichtig: Laden Sie bitte Ihre Lösung (.txt, .java oder .pdf) in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 23:59 (Mitternacht) ab.

[5] Sie erhalten von Ihrem Proseminarleiter eine E-Mail mit dem Zugangscode