FernUniversität in Hagen

FAKULTÄT FÜR
MATHEMATIK UND INFORMATIK

Lehrgebiet Programmiersysteme
Prof. Dr. Steimann

Fachpraktikum Programmiersysteme, WS 2011/12

Einstiegsaufgabe

Liebe Teilnehmerinnen und Teilnehmer des Fachpraktikums,

die folgende Aufgabe soll Ihnen als Vorbereitung für das Fachpraktikum dienen und sicherstellen, dass Sie mit einem vergleichbaren Kenntnisstand in die Veranstaltung gehen. Insbesondere kann Ihnen diese Aufgabe ein Gefühl für den Schwierigkeitsgrad des Fachpraktikums liefern. Während des Praktikums werden Sie mit drei bis vier Ihrer Kommilitonen in einem Team zusammenarbeiten, so dass ein Abbruch des Praktikums Ihre spätere Gruppe vor größere Probleme stellen könnte. Wir möchten sicherstellen, dass sich schon vor Antritt zur ersten Präsenzphase alle Teilnehmer über die Anforderungen im Klaren sind.

Bitte senden Sie Ihre Lösung nach erfolgreicher Praktikumsanmeldung spätestens am 11. September 2011 an `andreas.thies@fernuni-hagen.de`. Sollte bis dahin keine der Aufgabenstellung genügende Lösung eingegangen sein, gilt das Fachpraktikum für Sie als nicht bestanden. Auch wenn wir Absprachen und Gruppenarbeit unter den Teilnehmern nicht verhindern können, erwarten wir dennoch, dass jeder Teilnehmer eine individuelle Lösung einsendet.

Bei sich während der Bearbeitung auftuenden Fragen können Sie sich über die für dieses Praktikum eingerichtete Newsgroup `news://feu.informatik.kurs.1595.diskussion` mit Ihren Kommilitoninnen und Kommilitonen austauschen. Ebenso sind wir Betreuer natürlich auch gerne für Sie verfügbar, am einfachsten per Email, gerne aber auch per Telefon.

Ihr Fachpraktikumsteam,

Andreas Thies
Marcus Frenkel
Christian Kollee

Der Anfang: Einstieg in Eclipse

Machen Sie sich mit Eclipse vertraut![1]

- Installieren Sie sich eine aktuelle Version von Eclipse („Eclipse Classic") und erkunden Sie die Bedienungsweise. Die Begriffe wie *Workspace*, *Project*, *View*, *Perspective* und *Outline* sollten Sie sinnvoll in Zusammenhang bringen können.
- Schreiben Sie ein einfaches Java-Beispielprogramm und führen Sie es aus. Finden Sie heraus, wie der integrierte Debugger funktioniert und wie Sie Haltepunkte setzen können.
- Im Laufe der Fachpraktikums werden Sie Ihre Implementierung testen müssen. Erzeugen Sie ein paar kleine JUnit-Tests.[2] Erkunden Sie auch die in Eclipse eingebauten Assistenten, welche Ihnen das Grundgerüst für Ihre Tests weitgehend automatisch generieren können.[3]
- Erkunden Sie die in Eclipse eingebauten Refaktorisierungswerkzeuge. Machen Sie sich insbesondere mit dem Umbenennen von Variablen, Feldern und Methodennamen vertraut.
- Setzen Sie sich mit der SVN-Unterstützung in Eclipse auseinander. Laden Sie sich einen der frei verfügbaren SVN-Clients für Eclipse herunter und installieren Sie diesen.[4]
- Erzeugen Sie ein erstes Plugin für Eclipse. Auf der Eclipse-Homepage findet sich ein empfehlenswertes Tutorial hierzu.[5] Ihr Plugin sollten Sie als fertige jar-Datei exportieren und in Ihr eigenes Eclipse einbinden können.

---

[1] Soweit Sie noch gar keinen Kontakt mit Eclipse hatten, kann Ihnen das Tutorial von Michael Paap helfen, welches Sie auf seiner Homepage `http://feu.mpaap.de/eclipse/` finden. Ein englischsprachiges Texttutorial für die ersten Schritte mit Eclipse finden Sie unter `http://help.eclipse.org/helios/topic/org.eclipse.jdt.doc.user/gettingStarted/qs-BasicTutorial.htm`. Videotutorials zum Umgang mit Eclipse – insbesondere der Oberfläche und dem Debugger – finden Sie auf der Eclipse-Homepage unter `http://www.eclipse.org/resources/?category=Getting%20Started`. Bevorzugen Sie eine gedruckte Quelle, können wir Ihnen die ersten Kapitel von *Java-Entwicklung mit Eclipse: Anwendungen, Plugins und Rich Clients* von Berthold Daum empfehlen, wobei für einen ersten Einstieg auch die älteren Ausgaben hier noch ihren Zweck erfüllen sollten.

[2] Sollten Sie bisher nicht mit JUnit gearbeitet haben, empfiehlt sich das JUnit-Tutorial von Lars Vogel: `http://www.vogella.de/articles/JUnit/article.html`.

[3] Hinweise zur Unterstützung in Eclipse und weitere Beispiele hält ein gelungenes Tutorial der TU Darmstadt bereit: `http://www.mm.informatik.tu-darmstadt.de/courses/helpdesk/junit4.pdf`.

[4] Z.B. `http://subclipse.tigris.org/` oder `http://www.eclipse.org/subversive/`

[5] `http://www.eclipse.org/articles/Article-PDE-does-plugins/PDE-intro.html`

Motivation: Methodennamen in Eclipse

In der Programmiersprache Java folgt die Benennung von Programmelementen üblicherweise bestimmten Konventionen. So empfiehlt die Java Language Specification beispielsweise, Paket-, Methoden- und Variablennamen mit einem Kleinbuchstaben beginnen zu lassen, Typnamen hingegen durch einen Großbuchstaben einzuleiten. Auch wenn diese Richtlinien keine Überprüfung beispielsweise durch den Compiler erfahren, sollte man sie sich dennoch zu Herzen nehmen. Letztendlich macht man es damit nicht nur anderen Lesern einfacher, ein Programm zu erfassen, sondern kann auch manch technischem Problem aus dem Weg gehen. Benennt man z.B. ein Paket genau wie eine Klasse, kann man den Paketnamen nicht mehr referenzieren, wie das folgende nicht kompilierende Beispiel zeigt.[6]

```
package a;

public class a {
      a.a x;
}
```

Eine Konvention, die Namensgleichheit von Paketen und Typen ausschließt, verhindert solche Probleme schon im Ansatz.

Die Aufgabe: Typnamen in Eclipse

Im Rahmen dieser Aufgabe ist ein Eclipse-Plugin zu entwickeln, welches den Namen eines Typen (also Klasse, Interface oder Enum) überprüft und einen Vorschläg für eine Refaktorisierung macht. Aufgerufen wird dieses Plugin über das Kontextmenü des Package Explorer oder der Outline (siehe Abbildung 1).

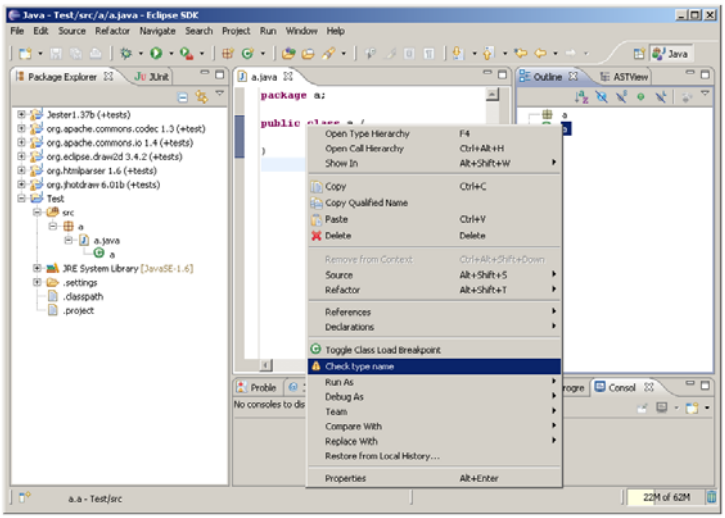

Abbildung 1 Das Plugin wird über das Kontextmenü der Outline oder des Package Explorer aufgerufen

Anschließend soll ihr Plugin den Namen des gewählten Typen dahingehend überprüfen, ob er mit einem Kleinbuchstaben beginnt. Ist dies der Fall, erfolgt ein entsprechender Hinweis (Abbildung 2) auf welchen hin bei einem Klick auf ‚Yes' ein Dialog geöffnet wird, welcher die Umbenennung des gewählten Typen erlaubt (Abbildung 3). Beachten Sie, dass dort bereits ein Vorschlag für einen geeigneten Bezeichner eingetragen sein soll, welcher den ersten Buchstaben des ursprünglichen Typnamen durch einen Großbuchstaben ersetzt.

Beginnt ein zu überprüfender Typname mit einem Großbuchstaben, so soll das Plugin dies lediglich mit einem Dialog bestätigen, welcher dann nur einen ‚OK'-Button aufweist (Abbildung 4).

---

[6] Sogenanntes *obscuring*, siehe The Java Language Specification, 3rd Edition, §6.3.2

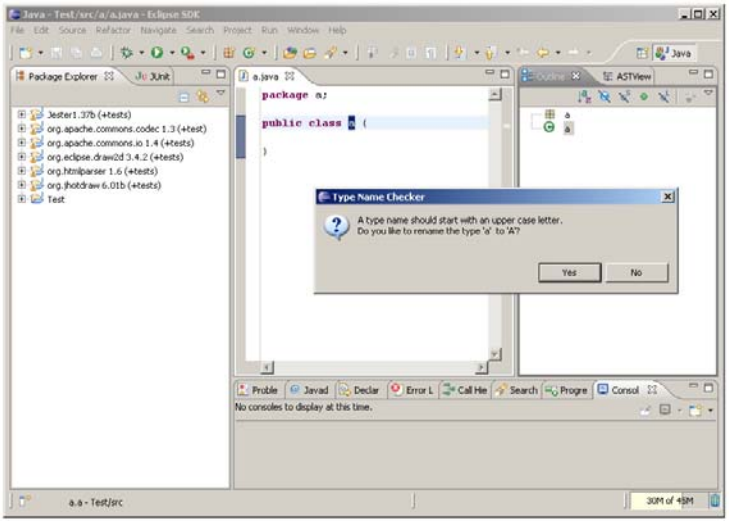

Abbildung 2 Beginnt der Typname mit einem Kleinbuchstaben, so soll eine Refaktorisierung vorgeschlagen werden

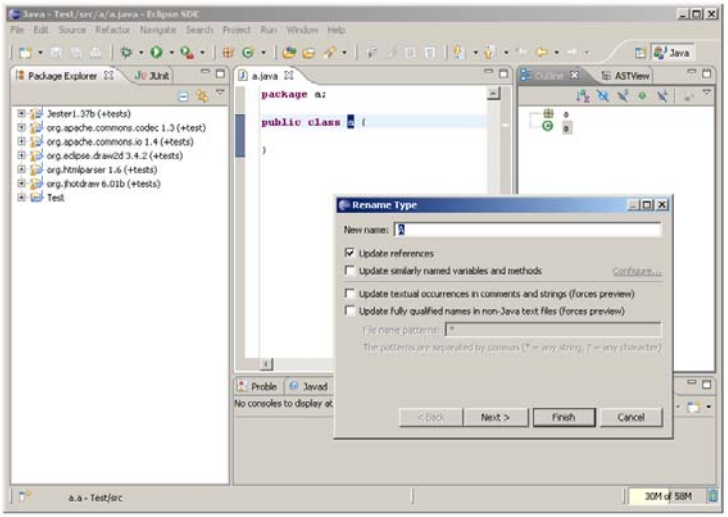

Abbildung 3 Der sich öffnende Refaktorisierungsdialog soll bereits einen Vorschlag für einen neuen Namen enthalten

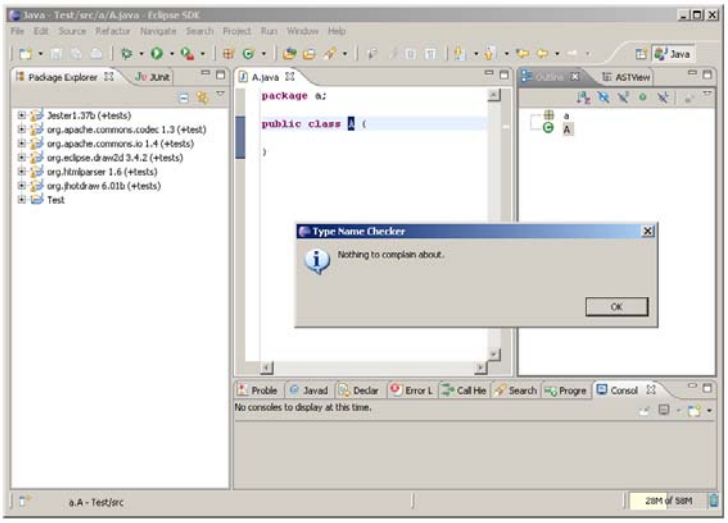

Abbildung 4 Ist der Typname nicht zu beanstanden, erhält der Benutzer einen entsprechenden Hinweis

Formales: Die Abgabe

Sobald Sie Ihr Plugin fertig gestellt haben, exportieren Sie es bitte als zip-Datei (File → Export → Archive File). Achten Sie bitte darauf, dass der Projektname Ihren Nachnamen enthält. Weiterhin tragen Sie bitte in der MANIFEST.MF des Plugins Ihren Namen und Ihre Matrikelnummer in dem Feld ‚Provider' ein. Die Zip-Datei ist spätestens am 11. September 2011 an `andreas.thies@fernuni-hagen.de` zu senden. Wir werden am Folgetag jede Abgabe kurz quittieren und im Laufe der folgenden Woche korrigieren. Abgaben vor dem 11. September haben gute Chancen, auch schon vorher korrigiert zu werden.

Hinweise: Tipps & Links

Wie bei der Nutzung eines jeden Frameworks kann man sich insbesondere bei der Entwicklung von Eclipse-Plugins viel Programmierarbeit ersparen, wenn man bereits vorhandene Funktionalität des Frameworks nutzt. Das Plugin in dieser Aufgabe erfordert nicht mehr als zwei Bildschirmseiten XML- und Java-Code. Die eigentliche Hauptaufgabe liegt in der Recherche, wie man sich die bereits in Eclipse vorhandene Funktionalität zu Nutze macht.

Eclipse selbst bietet dem Einsteiger Unterstützung bei den ersten Schritten, um ein eigenes Plugin zu schreiben. Erstellt man innerhalb von Eclipse ein neues Projekt (File → New → Project → Plug-in Project), kann man innerhalb des folgenden Assistenten aus verschiedenen Beispielen auswählen, an welchen man sich benötigte Funktionalitäten abschauen kann. Besonders hilfreich könnte Ihnen dabei das *Plug-in with a popup menu* sein, welches Ihnen zeigt, wie Einträge in ein Menü gemacht werden. Nebenbei können Sie dem Quellcode auch entnehmen, wie Sie dem Benutzer Popup-Fenster wie in Abbildung 4 präsentieren.

Eine meist gute Quelle für weiterführende Fragen bieten die oft mehr als ausführlichen, aber manchmal leicht veralteten Eclipse Corner Articles[7]. Im Artikel zum Eclipse AST[8] finden Sie einen Abschnitt zum Java-Modell, welches Eclipse nutzt, um Programmelemente wie Typen und Methoden abzubilden.

Aktuellere und versionsspezifische Informationen, welche aber oft knapper und nicht in Form eines Tutorials gehalten sind, hält die Eclipse Documentation[9] bereit. Wenn Sie schon wissen, wonach Sie suchen müssen,[10] ist die Eclipse Documentation Ihre erste Anlaufstelle.

Wissen Sie nicht, wonach genau Sie suchen müssen, kann Ihnen das am Lehrgebiet betriebene Eclipse-Wiki[11] weiterhelfen. Es lebt von den Beiträgen unserer Seminar-, Praktikums- und Abschlussarbeiter und kann Ihnen viele Fragen beantworten, die sich andere Studierende schon von Ihnen gestellt haben. Im vorhergehenden Praktikum war es Aufgabe, automatisch Refaktorisierungen anzuwenden. Eine Beschreibung, wie Sie die Refaktorisierungswerkzeuge und deren Dialoge intern ansprechen, ist dort mit Sicherheit zu finden. Sollten Sie im Eclipse-Wiki einmal nicht auf eine passende Antwort stoßen, sehen Sie es bitte als Ihre Pflicht, die gesuchte Information dort einzutragen sobald Sie an anderer Stelle Antwort gefunden haben.[12]

Sollten Fragen offen bleiben, helfen in einer Vielzahl der Fälle die durchaus empfehlenswerten Eclipse Community Forums[13]. Oft ergibt sich eine Antwort schon durch eine geeignete Suche, welche man übrigens der Netiquette halber unbedingt erschöpfend ausführen sollte, bevor man selbst eine Frage stellt.

---

[7] `http://www.eclipse.org/articles/`

[8] `http://www.eclipse.org/articles/article.php?file=Article-JavaCodeManipulation_AST/index.html`

[9] `http://help.eclipse.org/helios/index.jsp`

[10] Probieren Sie es aus, indem Sie einmal nach der Klasse ‚RenameSupport' suchen. Diese könnte Ihnen später dienlich sein...

[11] `http://wiki.fernuni-hagen.de/eclipse/index.php/Hauptseite`

[12] Einen Account erhalten Sie spätestens nach erfolgreicher Anmeldung.

[13] `http://www.eclipse.org/forums/`

Für Fleißige: Bonusaufgaben

Das in dieser Aufgabe zu entwickelnde Plugin ist zugegebenermaßen rudimentär. Gerne können Sie sich beliebige Erweiterungen ausdenken und implementieren. Hier ein paar Ideen:

- Trägt ein Typ den Namen ‚x', sollten Sie nicht ‚X' als Namen vorschlagen, wenn in der gleichen Datei (oder im gleichen Paket) schon ein weiterer Typ mit Namen ‚X' existiert. Führen Sie entsprechende Prüfungen durch.
- Das Aufrufen der Namensprüfung mittels Kontextmenü ist mehr als unpraktisch, wo Eclipse doch mit dem Konzept der Warnings auch anbietet, problematische Codestellen in bestimmten Farben zu kennzeichnen. Versehen Sie unpassende Typnamen mit entsprechenden gelb unterstrichenen Warnungen. Ein Eclipse Corner Article[14] zeigt Ihnen, wie das geht und auch das Eclipse-Wiki hat schon einen Artikel hierzu. Passend zu einer Warnung kann auch ein Quick-Fix angeboten werden; dieser sollte direkt den mit passenden Eingaben bestückten Refaktorisierungsdialog aufrufen.
- Neben Typen kann man wohl auch jede andere Art von Deklaration ungeschickt benennen. Testen Sie Paketnamen, Methoden und Variablen auf ihre Kleinschreibung. Finden Sie Methodennamen, die mit einem ‚get' beginnen, obwohl sie keinen Rückgabewert liefern. Ähnlich können Sie mit den Präfixen ‚set', ‚is' und ‚has' verfahren.

[14] `http://www.eclipse.org/articles/Article-Mark%20My%20Words/mark-my-words.html`