Hinweise zu Eclise

Überblick:

1. Vorbermerkungen
2. Einstieg in Eclipse
3. Importieren des Praktikumsprojekts
4. Screenshots von ein paar wichtigen Aktionen

1. Vorbemerkungen

Es gibt keine perfekte Welt. Ein Folge davon ist, dass Sie auf unterschiedlichen Computern unterschiedliche Softwareversionen, unterschiedliche Betriebssysteme und unterschiedliche Einstellungen vorfinden. Einige dieser Punkte können Sie beeinflussen, andere müssen Sie einfach hinnehmen und im Einzelfall beachten.

Java-Entwicklungsumgebung

Während Eclipse mit der praktisch auf jedem Rechner vorinstallierten JRE (Runtime-Environment) zurecht kommt, sollten Sie unbedingt die JDK (Development-Kit) installieren. Nur dann haben Sie Zugriff zu Javadoc, zu den Quelltexten der Bibliothek (src.zip) und zu weiteren nützlichen Werkzeugen.

In den Praktikumsräumen ist Java 5 installiert. Durch einfache Änderung des Suchpfades (PATH) können Sie Ihr Java auf die Version 6 upgraden. Sie haben dann ein paar (wenige) Probleme weniger

Zu Hause haben Sie vermutlich Java 7 oder bereits Java 8 installiert. Diese beiden Java-Versionen unterscheiden sich erheblich von Java 5/6. Dies müssen Sie bei der Bearbeitung Ihrer Praktikumsaufgaben beachten. .

Eclipse

Neuere Eclipse-Versionen haben einige Vorteile für die Java-Entwicklung. Dies macht sich natürlich nur bemerkbar, wenn man sich auch auskennt.

Im Praktikum ist Eclipse 3.5 installiert. Sie haben auf Ihrem Rechner vermutlich 3.9. JUnit ist in den Versionen 4.5 bzw. 4.11 installiert. Ich nehme Rücksicht auf die Unterschiede, indem ich die Vorlagen anpasse.

Zeichenkodierung

Sie kennen den ASCII-Zeichensatz, wissen vermutlich aber auch, dass dieser für praktische Anwenungen (z.B. Umlaute) zu wenige Zeichen hat. Dementsprechend gibt es (leider unterschiedliche) Erweiterungen. Auf Windows-Systemen und im Praktikum ist die Kodierung ISO 8859-1 eingestellt. Auf Mac- und Linux-Systemen verwendet man den

Standard UTF8. Diese Unterschiede führen dazu, dass beim Übertragen von dem einen zum anderen System Umlaute falsch dargestellt werden.

Es gibt zwei praktikable Lösungen

a) Sie vermeiden Umlaute (ich versuche bei den Praktikumsaufgaben daran zu denken).

b) Sie verwenden in allen Eclipse-Umgebungen die gleiche Einstellung (empfehlenswert ist UTF8).

Eingestellt wird die Kodierung allgemein in **`Windows` → `Preferences` → `General` → `Workspace`** oder pro Projekt in **`Properties` → `Ressource`**.

Übertragen der Aufgaben Praktikum – eigener Rechner.

Hierzu können Sie ein sftp-Protokoll Nutzen (Linux: Datei Explorer, Windows winscp). Aber bitte beachten: Kopieren Sie keine Dateien in die Eclipse-Workspaces hinein! Sie machen sich nur Probleme. Wählen Sie statt dessen den Weg über Export/Import. Wenn Sie das geschickt **machen**, können Sie Projekte mit allen Einstellungen fertig übertragen.

2.Einstieg in Eclipse

Eine moderne IDE - wie Eclipse - kann stark bei der Lösung der Praktikumsaufgaben und auch bei dem Erlernen von Java sehr behilflich sein. Der Einarbeitungsaufwand zahlt sich aus.

Hinweise gibt es auch in der Vorlesung und im Skript!

Mark Dexter, Eclipse and Java for total Beginners
http://www.youtube.com/watch?v=xO5DpU2j-WE

Mark Dexter, Eclipse and Java: Using the Debugger
http://www.youtube.com/watch?v=WeSitNPAExg

Die Videos gibt es auch zum direkten Download:
http://www.eclipse.org/resources/?category=Getting%20Started

Die Klassen zu den Praktikumsaufgaben finden Sie auf der AP2-Seite unter www.gm.fh-koeln.de/ehses in der Datei *vorlage1.zip*. Dieses Archiv enthält ein fertiges Eclipse-Projekt. Sie sollen das Archiv nur auf den Rechner kopieren, aber nicht auspacken!

3. Importieren in Eclipse

`File → Import… → Existing Projects into Workspace`

`dann` **`select archive file`** `auswählen.`

Das Projekt wird mit allen Einstellungen automatisch erzeugt. Das Projekt sollte keine Compilerfehler enthalten. Wenn das der Fall ist, oder auch nur mal zur Kontrolle, sollten Sie die Projekteinstellungen überprüfen.

Wählen Sie im **Package Explorer** das Projekt mit der rechten Maustaste und rufen Sie dann „Properties“ auf. Überprüfen Sie, ob als „Java-Compiler“ die richtige Version angegeben ist. Und dann überprüfen Sie die Pfadeinstellungen (**`Java Build Path`**). Hier sind wiederum die Einstellungen unter **`Source`** und unter **Libraries** wichtig.

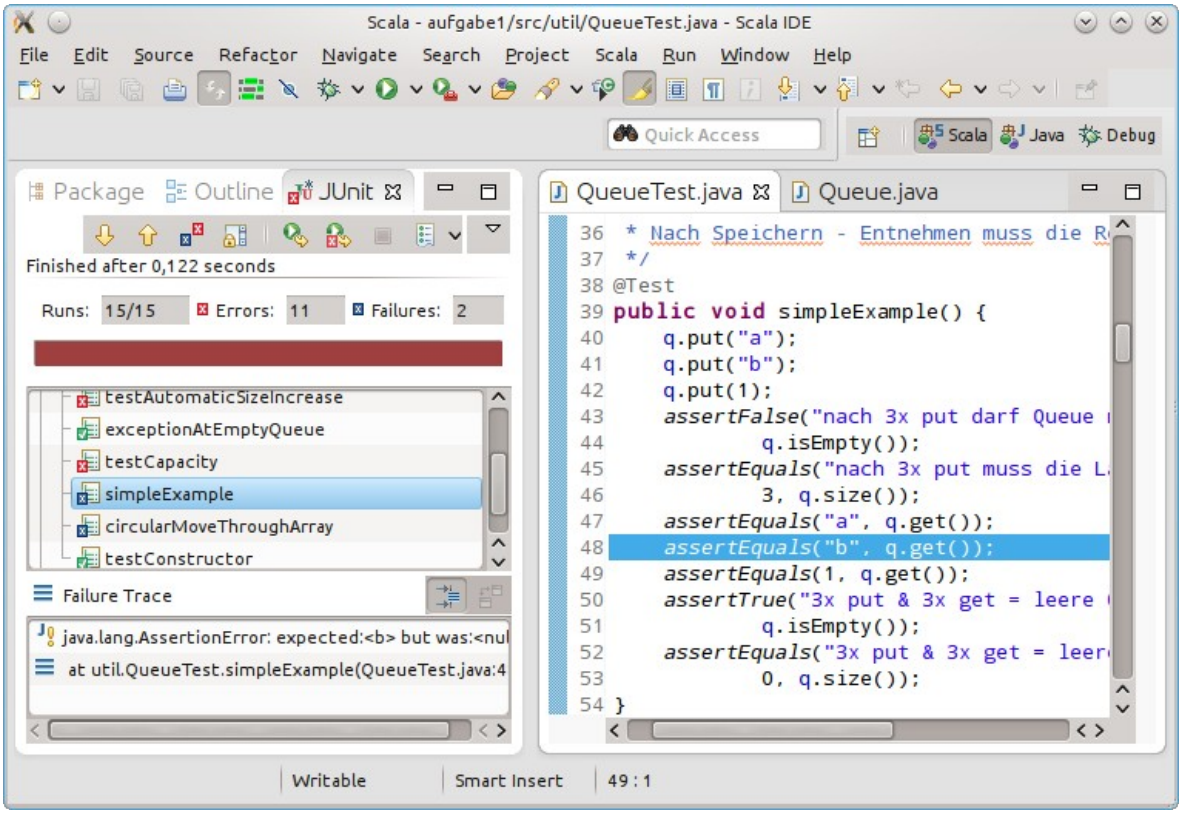
Wenn das Projekt keine Compilerfehler zeigt, sollten Sie eine Kopie des Projekts erstellen (Copy-Paste). Damit können Sie später Ihre Änderungen nachvollziehen (Package-Explorer:

Die Aktion Compare With des Package Explorer Kontextmenüs zeigt die Unterschiede an. (Auswählen zweier Verzeichnisse oder Dateien erfolgt mit der linke Maustaste bei gedrückter STRG-Taste),

Hinweis: Es ist oft sinnvoll, die neuen Änderungen mit dem alten Zustand zu vergleichen. **Lassen Sie nicht alten Code in der neuen Fassung stehen!**

4. Screenshots

Die folgenden Beispiele sind im Skript ausführlicher erläutert!



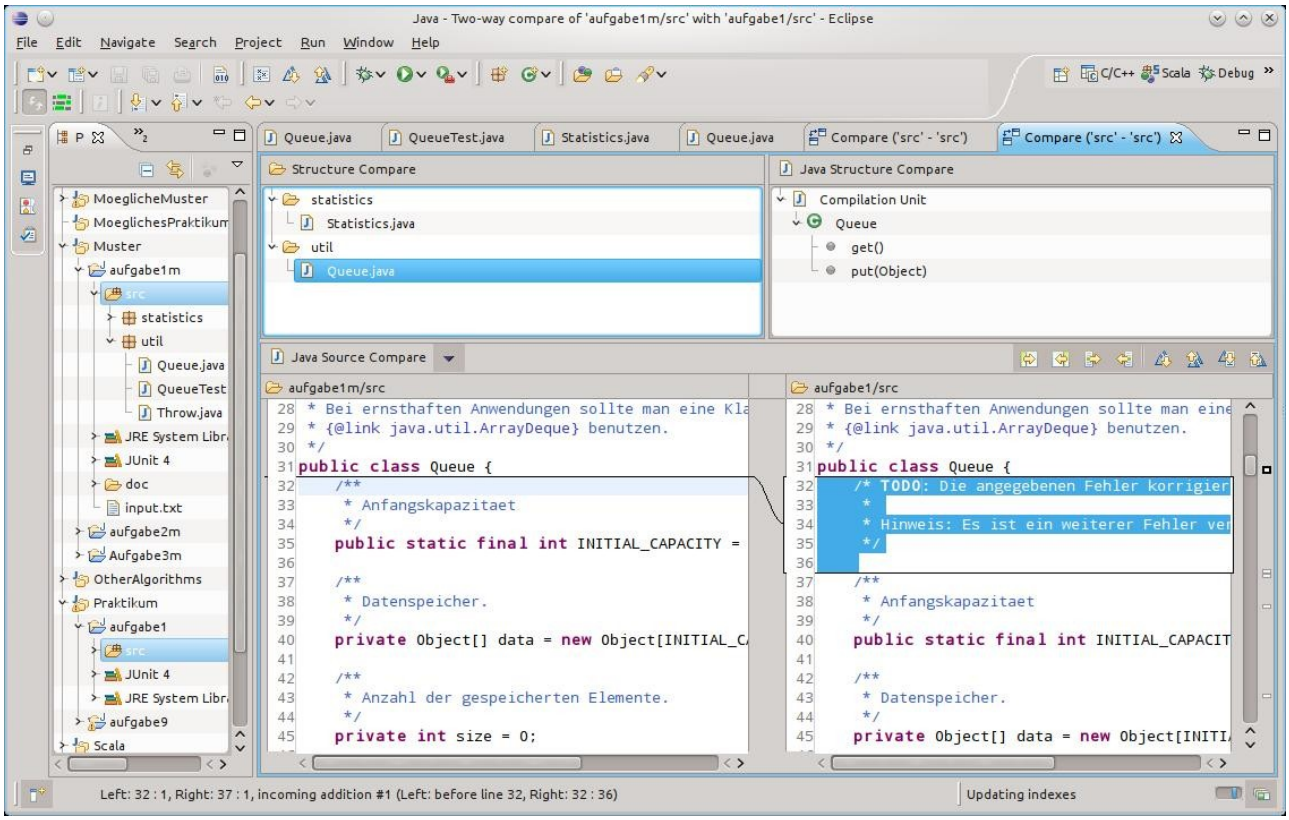
JUnit zeigt in dem linken Fenster die Testergebnisse an. Bei fehlgeschlagenen Teste ist die Testmeldung mit der Stelle verlinkt, wo der Fehler erkannt wurde. Außerdem ist eine kurze Meldung angegeben, Hier wurde von `q.size()` eine 1 erwartet, das Ergebnis war aber 0.

Grüm markierte Tests waren erfolgreich.

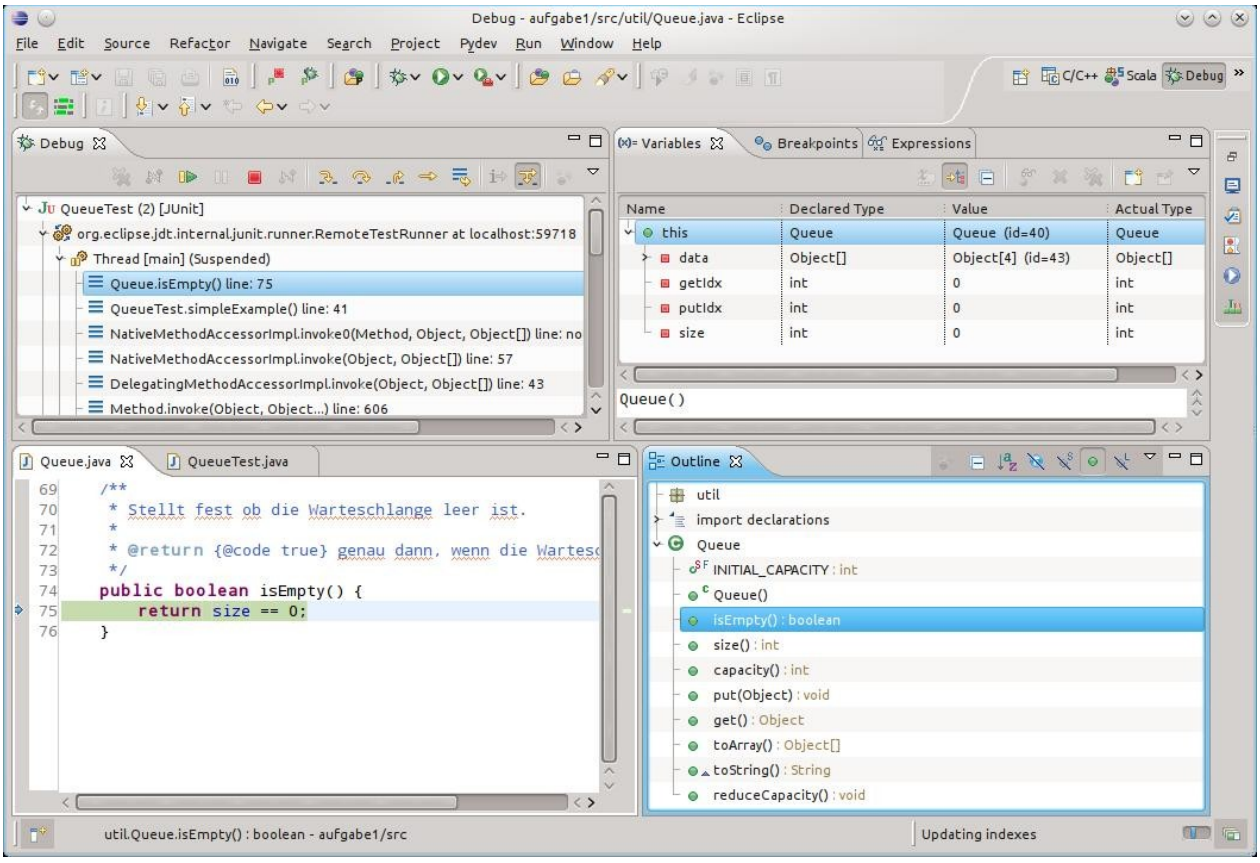
Blau markierte Tests lieferten nicht das erwartete Ergebnis.

Rot Markierte Tests zeigten einen Laufzeitfehler (Exception).

Auch bei Exceptions ist eine Verknüpfung zum Quelltext angegeben.



Hier sind die Unterschiede zwischen zwei Dateien (Musterlösung mit Vorlage der Aufgabe) hervorgehoben. In der Übersicht ist erkennbar in welchen Methoden weitere Unterschiede bestehen.



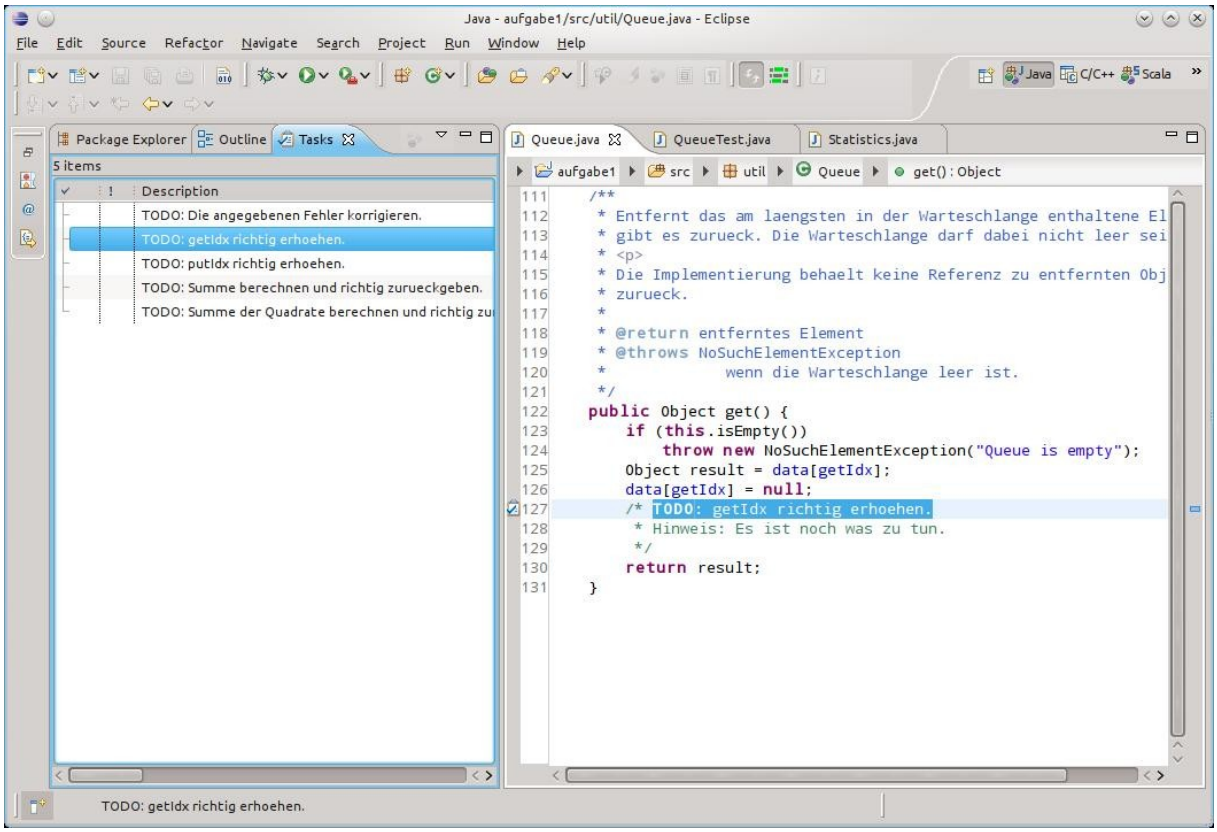
Der Debugger hat an einem Haltepunkt angehalten. In dem Fenster *Debug* sieht, man, von wo die Methode `isEmpty` aufgerufen wurde. Der Balken über dem Fenster steuert den weiteren Ablauf (*continue, stop, step into, step over, step out, goto cursor, drop frame*).

Das Fenster *Variables* zeigt die aktuellen Variableninhalte. Objekte lassen sich aufklappen. Das Fenster lässt sich konfigurieren (nach unten zeigendes Dreieck am Fensterrand). Man kann dort auch erkennen, wenn der deklarierte Typ vom tatsächlichen Typ abweicht.

Hier ist das leider nicht zu sehen. Wenn man aber `data` aufklappt sieht man dass die Inhaltswerte den Typ `Double` haben.

Man kann auch Haltepunkte für Ausnahmen definieren. Dann hält das Programm unmittelbar an der Fehlerstelle an, so dass man sich in aller Ruhe die aktuellen Variableninhalte anschauen kann.

Der *Variables*-Reiter lässt sich in *Breakpoints* umschalten. Hier kann man die Haltepunkte verwalten. Z.B. kann man festlegen, dass ein Haltepunkt nur bei bestimmten Bedingungen wirksam wird.



Wenn Sie in vorhandenen Java-Dateien etwas tun sollen, markiere ich diese Stellen durch TODO – Kommentare. Das Fenster „Tasks“ gibt Ihnen einen Überblick über die noch nicht erledigten Aufgaben.

Natürlich ist das nur dann hilfreich, wenn man die TODO-Markierungen auch wieder entfernt (s. oben: keinen Müll zurücklassen!)