


Ruby Refactoring Plug-In für Eclipse



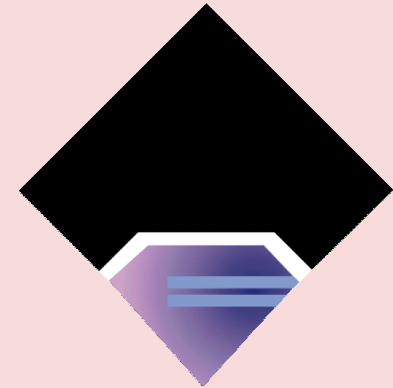
Team: Lukas Felber
Thomas Corbat
Mirko Stocker

Betreuung: Prof. Peter Sommerlad
Experte: Dr. Dirk Bäumer

Übersicht

- Einführung
- Studienarbeit
- Refactorings
 - Pro Refactoring:
 - Demo
 - Erläuterungen
- Testen
- Fragen

- Ruby Development Tools
- IDE basierend auf Eclipse
- Grundlage von RadRails
- Features wie:
 - TestUnit Runner
 - Debugging
 - Code Completion



**Ruby Development Tools
Refactoring Support**

Einführung - JRuby

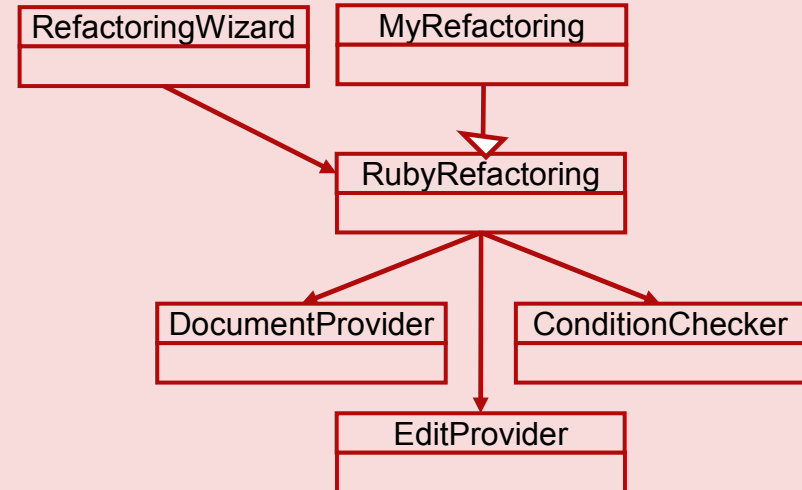
- Ruby Implementierung in Java
- Stellt einen AST zur Verfügung
 - ohne Kommentare
 - optimiert für Interpretation

- Comment Handling
- Positions-Fehler
- AST-Rewriter
- Basis für Refactorings

- Code Generators:
 - Generate Accessors
 - Generate Constructors Using Fields
 - Override Method
- Refactorings:
 - Push Down Method
 - Rename Local Variable

Refactoring Basis Komponenten

- Refactoring Wizard
- Ruby Refactoring
- Edit Provider
- Document Provider
- Condition Checker





Refactorings

- Convert Local Variable to Field
- Encapsulate Field
- Extract Method
- Move Method
- Move Field
- Split Temporary Variable
- Inline Method
- Inline Class
- Inline Temp
- Merge Class Parts
- Rename Field
- Rename Method
- Rename Class
- Rename Local Variable

Convert Local Variable to Field



- Alle vorkommen der angewählten lokalen Variable finden
- Falls gewünscht Deklaration der lokalen Variable entfernen und Deklaration des Feldes in Konstruktor einfügen.
- Lokale Variablen durch Feld ersetzen

Encapsulate Field

- Accessors für selektiertes Feld finden
- Entsprechende getter- und setter-Methoden generieren
- Bestehende Accessors löschen

Ruby Accessors:

```
attr_accessor :myPublicField  
attr_reader   :myReadableField  
attr_writer  :myWritableField
```

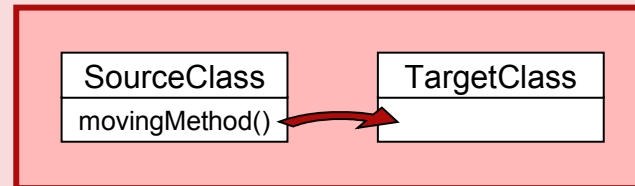
Accessors ermöglichen den Zugriff auf Felder von ausserhalb einer Klasse.



Extract Method

- Falls nötig aktuelle Selektion sinnvoll erweitern
- Selektierten Codeblock ermitteln
- Lokale Variablen, welche als Argumente benötigt werden, ermitteln.
- Rückgabewerte ermitteln (in Ruby auch mehrere möglich).
- Selektierter Codeblock mit Methodenaufruf ersetzen und neue Methodendefinition einfügen

Move Method



- *movingMethod* ermitteln und verschieben
- Falls verlangt delegate-Method in *SourceClass* einfügen
- Referenzen anpassen:
 - Aufrufe von *movingMethod* in *SourceClass*
 - Aufrufe von *SourceClass* Methoden in *movingMethod*
 - Zugriffe auf Felder von *SourceClass* in *movingMethod*
- Accessoren und Methoden-Visibility anpassen, dass alle Referenzen gültig bleiben



Move Field

- Feld in eine andere Klasse verschieben:
 - Accessors in Zielklasse erstellen, in Quellklasse durch Delegate-Methode ersetzen
 - Verwendung innerhalb der Quellklasse auf Zielklasse delegieren

Split Temporary Variable

- Jeder Verwendung einer lokalen Variablen einen neuen Namen zuweisen
- Prinzipiell wie Rename Local Variable, nur in anderem Scope

Inline Method

- Ersetzt Methodenaufruf durch Definition
 - Parameter umbenennen oder neu einführen
 - Aufrufe von self auf Referenz umbiegen
 - returns entfernen und Werte zuweisen
 - auf Wunsch Definition löschen
- Bedingungen:
 - Höchstens ein return-Aufruf
 - return muss am Ende stehen

Inline Class

- Integrieren der Funktionalität einer Klasse in eine andere.
- Restriktion:
 - Aufnehmende Klasse muss eine Instanz der Ziel-Klasse in ihrem Konstruktor anlegen.
 - Dieser Aufruf des nicht mehr existierenden Konstruktors wird durch die eigene Instanz ersetzt.
- Ablauf:
 - Auswahl der Zielklasse
 - Verschieben der Methoden und Felder
 - Umbenennen der Methoden und Felder bei Konflikten
 - Verschieben und Umbenennen des Konstruktors
 - Löschen der integrierten Klasse

Inline Temp

- Ersetzt eine lokale Variable durch deren Wert

Code:

```
summe = 5 + 5  
produkt = summe * 5
```

produkt hat den Wert 50

Nach Ersetzung:

```
produkt = 5 + 5 * 5
```

produkt hat den Wert 30

Besser wäre:

```
produkt = (5 + 5) * 5
```

produkt: 50

– keine Parameter

- Problem:
 - Veränderung von mathematischen Ausdrücken

Merge Class Parts

- Zusammenziehen von Klassenteilen
- Zwei Refactorings für Teile in derselben Datei und Teile in mehreren Dateien
- Ablauf:
 - Selektion der Klassen-Teile
 - Auswahl der Ziel-Definition
 - Verschieben der Teile ans Ende der Ziel-Definition

Rename Field

- Umbenennen eines Feldes
- Vorgehen:
 - Der Cursor muss auf einem Feld platziert werden
 - Eingabe eines neuen gültigen Namens
 - Vorschlag der umzubenennenden Vorkommen
 - Auswahl der umzubenennenden Vorkommen
 - Umbenennen der Accessors

Rename Method

- Umbenennen einer Methode
- Vorgehen:
 - Selektion einer Methoden-Definition
 - Eingabe eines neuen gültigen Namens
 - Vorschlag der umzubenenennenden Aufrufe
 - Auswahl der umzubenenennenden Aufrufe



Rename Class

- Klasse umbenennen:
 - Instanziierungen
 - andere Definitonen derselben Klasse
 - Subklassen

Rename Local Variable

- Lokale Variablen beinhaltet:
 - “normale”, lokale Variablen
 - Methodenargumente
 - Variablen in Blöcken, |foo, bar|

```
def method argument, arg_with_def = "default", *rest_args, &block_arg
  block = block_arg
  rest_args.each do |arg|
    p arg
  end
end
```

- JUnit-Tests
- File Driven Tests
 - Dateien für Original und erwartetem Resultat
 - Konfiguration für Refactoring in Property-File
 - Keine Änderungen an Java-Code für neue Tests
- Continuous Integration mit Cruise Control

Ausblick

- Integration in RDT / JRuby
- Weiterentwicklung
 - Weiter Refactorings
 - Adaption an neue Versionen (Ruby/JRuby/RDT)
 - Erweiterung bestehender Refactorings
- Comment Handling



Ende



Fragen