Programmieren in Java

http://proglang.informatik.uni-freiburg.de/teaching/java/2015/

Java-Übung Blatt 7 (Visitor-Pattern)

2015-06-15

Hinweise

- Ändern Sie nicht die Schreibweise von Bezeichnen, die auf dem Übungsblatt vorgegeben sind. Dies betrifft sämtliche global sichtbaren Namen von Eclipse-Projekten, Paketen, Klassen, etc.
- Bezeichner und Kommentare bitte auf *Englisch*!
- Schreiben Sie *sinnvolle* Kommentare.
- Laden Sie Ihre Lösungen mit Subversion (SVN) ins Übungssystem hoch. Den entsprechenden Pfad finden Sie online.
- Sollte das Übungssystem Ihre Einreichung nicht übersetzen können, dann wird sie nicht korrigiert und Sie erhalten keine Punkte.
- Die Korrektur Ihrer Abgabe wird unter dem Namen `Feedback-<login>-ex<NN>_<X>.txt` in das jeweilige Projektverzeichnis eingecheckt. Dabei ist `<login>` Ihr *myAccount*-Name und `<NN>_<X>` der Projektname.
- Sie können Ihre Gesamtpunktzahl im Übungsportal einsehen.
- Dokumentieren Sie Ihren Zeitaufwand, den gefühlten Schwierigkeitsgrad, sowie Probleme beim Lösen der Aufgabe oder auch Anregungen und Vorschläge zur Verbesserung in der Datei `<Projektverzeichnis>/erfahrungen.txt`

Hinweis: Dieses PDF stammt aus dem Archiv `ex07.zip`. Dieses Archiv enthält weiterhin noch Skelett-Projekte, die für die Bearbeitung der Aufgaben hilfreich sind.

Exercise 1 (PrettyPrinter (5 + 1 Punkte))
Projekt: `ex07_1`. Package: `expr.operations`.
Benutzen Sie das mitgelieferte Skelett-Projekt zu dieser Aufgabe. Es enthält das Datenmodell für arithmetische Ausdrücke mit Variablen im Package `expr`. Ein Ausdruck hat eine der folgenden Formen:

- eine Konstante, z.B. 42;
- eine Variable, z.B. $\$x$: Variablen haben einen Namen (ein beliebiger String, wie „x“) und werden zusätzlich mit einem vorangestellten Dollarzeichen („$“) gekennzeichnet;
- eine Addition, z.B. $(\$x + 42)$;
- ein Produkt, z.B. $(\$y * (\$z + 42))$.

Im Projekt ist auch das Interface `IExprVisitor<T>` im Package `expr` definiert. Alle Ausdrucks-Klassen unterstützen bereits eine `accept`-Methode für `IExprVisitor<T>`. In der Package `expr.operations`, in der auch Sie arbeiten sollen, befindet sich zur Anschauung bereits eine modifizierte Version des `SizeVisitor`s aus der Vorlesung.

Implementieren Sie mit dem Visitor Pattern die im folgenden beschriebenen zwei Operationen, die einen Arithmetischen Ausdruck als `String` formatieren (so genannte *Pretty-Printer*):

1. Einen algebraischen Pretty-Printer, der Ausdrücke vollständig geklammert mit den üblichen Infix-Operatoren für Addition („+“) und Multiplikation („*“). Beispiel: `new Add(new Produkt(new Var("x"), new Const(2)), new Const(3))` wird zu `"(($x * 2) + 3)"` formatiert.
2. Einen Pretty-Printer für *Umgekehrte Polnische Notation*[1]. Beispiel: `new Add(new Produkt(new Var("x"), new Const(2)), new Const(3))` wird zu `"$x 2 * 3 +"` formatiert.

Implementieren Sie außerdem folgende weitere Operation mit dem Visitor Pattern.

3. Einen Visitor, der arithmetische Ausdrücke in äquivalente Ausdrücke (`IExpr`) mit rechtsassoziativen Additions- und Produkt-Operatoren umformt.
   Beispiel: Der Ausdruck $((((\$x + 2) + 3) * 2) * 3)$ wird zu $(((\$x + (2 + 3)) * (2 * 3))$.

Beachten Sie bei Punkt 1 und Punkt 2 die korrekte Ausgabe von Variablen.
Bei Punkt 3 dürfen Sie *innerhalb Ihrer Visitor-Implementierung* `Object.getClass()` oder `instanceof` benutzen, welche Sie bereits aus der Vorlesung (Implementierung von `equals`) kennen, um zu überprüfen, ob ein arithmetischer Ausdruck eine Addition bzw. ein Produkt ist.
Bonusaufgabe (1 Punkt) Vermeiden Sie o.g. `Object.getClass()/instanceof`-Test.
Testen Sie ihre Implementierung mit JUnit.

[1] `http://de.wikipedia.org/wiki/Umgekehrte_Polnische_Notation`

Exercise 2 (Boolesche Ausdrücke (5 Punkte))
Projekt: `ex07_2`. Package: `bexpr`.

1. Entwickeln Sie, analog zu den arithmetischen Ausdrücken aus der Vorlesung, ein Datenmodell für *Boolesche Ausdrücke*. Ein Boolescher Ausdruck ist
   - eine *Wahrheitswert*, d.h. **true** oder **false**
   - eine *Variable*
   - eine *Negation* eines Booleschen Ausdrucks
   - eine *Konjuktion* von zwei Booleschen Ausdrücken („und"-Verknüpfung)
   - eine *Disjunktion* von zwei Booleschen Ausdrücken („oder"-Verknüpfung)

   Ihr Datenmodell soll ein entsprechendes Visitor-Interface unterstützen.

2. Implementieren Sie eine Operation zum Auswerten von Booleschen Ausdrücken als Visitor.

   Hinweis: Falls Sie eine Eingebung für die Auswertung der Variablen brauchen, lassen Sie sich vom Code aus der Vorlesung[2] inspirieren.

3. Testen Sie Ihren Visitor zum Auswerten von Booleschen Ausdrücken für wesentliche Fälle mit JUnit.

[2] `https://github.com/proglang/JavaCourse/tree/master/src/lecture20150615/visitor`