Übung 03: Dynamische Bindung

Abgabetermin: 7.4.2016, 10:00

Name: _____ **Matrikelnummer**: _____

Informatik: ☐ G1 (Grimmer) ☐ G2 (Prähofer) ☐ G3 (Prähofer)

WIN: ☐ G1 (Khalil) ☐ G2 (Hummel) ☐ G3 (Khalil)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 3	24	Java-Programm Ausgabe eines Testlaufs	Java-Programm	☐	

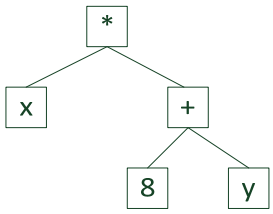
Übung 03: Expression Trees (24 Punkte)

Implementieren Sie ein Klassensystem für die Repräsentation, Berechnung und Vereinfachung mathematischer Ausdrücke. Die binären Operatoren Addition, Multiplikation, Subtraktion und sowie konstante Ausdrücke und Variablen sollen dabei als eigene Klassen implementiert werden. Die binären Operatoren speichern jeweils ihren linken und rechten Operanden, die wieder beliebige Ausdrücke sein können. Konstante haben einen `double`-Wert und Variablen einen Namen und einen veränderlichen `double`-Wert, den man setzen kann.

Eine Expression kann man dann z.B. folgend erzeugen:

```
Expr exp = new Mul(new Var("x"), new Add(new Const(8), new Var("y")));
```

und repräsentiert dann folgenden Expression-Tree:



Ausdrücke sollen nun folgende Operationen implementieren:

- `double evaluate()` – berechnet den double-Wert des Ausdrucks. Die Methode wirft bei einer Division durch 0 eine ArithmeticException.
- `String toInfixString()` – Erzeugt eine Infix-Notation des Ausdrucks, wobei 2-stellige Operationen mit Klammern ausgegeben werden sollen
  - z.B. `exp.toInfixString()` soll den String `"(x * (8.0 + y))"` liefern
- `Expr simplify()` soll den Ausdruck nach den unten stehenden Regeln vereinfachen und den vereinfachten Ausdruck als Ergebnis liefern.

Implementieren Sie des Weiteren `equals`-Methoden für die Expressions (Sie können sich die equals-Methoden durch Eclipse generieren lassen) und eine Methode

- `boolean isEquivTo(Expr other)`

die Äquivalenz von Ausdrücken feststellt. Äquivalenz ist folgend definiert:

- Konstante und Variablen sind äquivalent, wenn sie gleich (`equals`) sind.
- Zwei Additionen und zwei Multiplikationen sind äquivalent, wenn die beiden linken und die beiden rechten Ausdrücke äquivalent sind oder der rechte mit dem linken und der linke mit dem rechten Ausdruck äquivalent ist (Kommuntativität von + und *).

- Zwei Subtraktionen und zwei Divisionen sind äquivalent, wenn die beiden linken und die beiden rechten Ausdrücke äquivalent sind.

Vereinfachungsregeln:

Die Vereinfachung soll folgende Regeln berücksichtigen

- $a + 0 = a$
- $0 + a = a$
- $a + a' = 2 * a$ a äquivalent a'
- $a - 0 = a$
- $a - a' = 0$ a äquivalent a'
- $a * 0 = 0$
- $0 * a = 0$
- $a * 1 = a$
- $1 * a = a$
- $0 / a = 0$
- $a / 1 = a$
- $a / a' = 1$ a äquivalent a'

Hinweise:

- Die Basisklasse `Expr` ist eine abstrakte Klasse, die die entsprechenden Methoden für die Expressions deklariert. Überlegen Sie, welche Methoden und Klassen abstrakt sind, und welche Sichtbarkeitsattribute sinnvoll sind.
- Implementieren Sie dann Unterklassen von `Expr` zur Repräsentation von Kontanten, Variablen, Addition, Multiplikation, Division und Subtraktion. Führen Sie, wenn sinnvoll, Zwischenklassen ein.
- Die Methoden müssen rekursiv programmiert werden. Achten Sie darauf, dass Sie bei der Vereinfachung von Ausdrücken zuerst die Unterausdrücke rekursiv vereinfachen.
- Ausdrücke sollen bei der Vereinfachung nicht evaluiert werden.
- Sie müssen bei der Vereinfachung oft prüfen, ob ein Audruck eine Konstante 0 oder 1 ist. Dies müsste man ohne zusätzliche Methoden aufwendig folgend umsetzen:

  ```
  if (expr instanceof Const && ((Const) expr).getValue() == 0) ...
  ```

  Führen Sie daher Methoden

  ```
  boolean isZero()
  boolean isOne()
  ```

  ein, die diese Prüfungen entsprechend kapseln.
- Variablen mit gleichen Namen sollen immer durch dasselbe Objekt repräsentiert sein und damit denselben Wert haben. Überlegen Sie sich einen Mechanismus, um das zu erreichen.

Dialog zum Arbeiten mit Expresssions

Bereitgestellt ist eine interaktive Anwendung zum Arbeiten mit Ausdrücken. Folgend ist ein Beispieldialog angeben. Die Operationen erlauben das Anlegen von Konstanten und Variablen. Diese Objekte werden unter einem Index abgespeichert. Für binäre Ausdrücke können dann linker und rechter Ausdruck aus den bereits abgelegten Ausdrücken ausgewählt werden und der binäre Ausdruck wird wieder zur Verwendung abgelegt. In gleicher Weise funktionen die Evaluierung und die Vereinfachung. Mit einem Assignment wird einer Variablen ein Wert zugewiesen.

In der Programmvorgabe ist diese interaktive Anwendunung bereits vorbereitet. Sie müssen aber an den mit TODO gekennzeichneten Stellen, die entsprechenden Ergänzungen durchführen.

Testen Sie Ihre Implementierung mit dieser interaktiven Anwendung.

```
Expression Calculator Commands
==============================
c - new constant
v - new variable
+ - new add expression
- - new sub expression
* - new mult expression
/ - new div expression
a - assign value to variable
e - evaluate expression
s - simplify expression
q - quit
==============================
Command: c
Const value: 1
  [0]: 1.0
Command: v
Variable name: x
  [1]: x
Command: a
Variable name: x
Value: 2
Command: +
Expression index for left: 0
Expression index for right: 1
  [2]: (1.0 + x)
Command: +
Expression index for left: 1
Expression index for right: 0
  [3]: (x + 1.0)
Command: +
Expression index for left: 2
Expression index for right: 3
  [4]: ((1.0 + x) + (x + 1.0))
Command: s
Expression index for simplification: 4
  [5]: (2.0 * (1.0 + x))
Command: -
Expression index for left: 2
Expression index for right: 3
  [6]: ((1.0 + x) - (x + 1.0))
Command: s
Expression index for simplification: 6
  [7]: 0.0
Command: q
```