

Übung 09: OOP

Abgabetermin: 13. 1. 2009

Name: _____

Matrikelnummer: _____

Gruppe: G1 (Prähofer) G2 (Wolfinger) G3 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Pkte
Aufgabe 09.1	12	<input type="checkbox"/>	Java-Programm Testplan Testergebnisse	Java-Programm	<input type="checkbox"/>	
Aufgabe 09.2	12	<input type="checkbox"/>	Java-Programm Testplan Testergebnisse	Java-Programm	<input type="checkbox"/>	

Aufgabe 9.1: Expressions

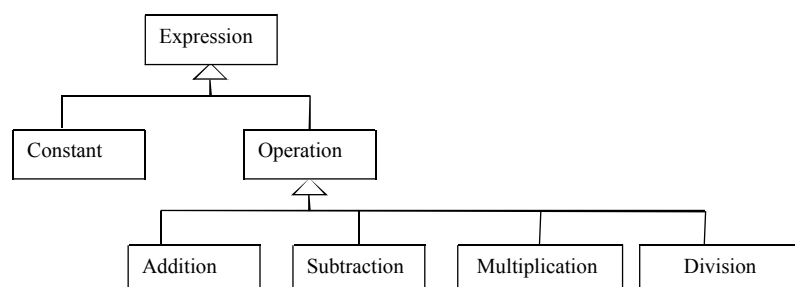
Wir wollen ein Klassensystem für arithmetische Ausdrücke (*Expression*) realisieren. Expressions können sein:

- *Constant*: hat ein *int*-Feld, welches den Wert speichert
- *Operation* hat einen linken und rechten Unterausdruck
Es gibt folgende Untertypen
 - *Addition*
 - *Subtraction*
 - *Multiplication* und
 - *Division*

Realisieren Sie Operationen für das Berechnen der Expression (*int evaluate()*) und für das Erzeugen einer String-Repräsentation (*String toString()*). Realisieren Sie entsprechende Konstruktoren für die Klassen. Testen Sie Ihr Programm, indem Sie mehrere Expressions aufbauen und *evaluate* und *print* ausführen.

Hinweise:

Anbei die Klassenhierarchie für das Beispiel:



Gehen Sie bei der Lösung folgend vor:

1. Implementieren Sie die Klassen und die Methoden in Java
2. Stellen Sie einen Testplan auf und testen Sie das Programm

Aufgabe 08.2: Einlesen und Rechnen mit Expressions

Implementieren Sie nun eine Methode

```
static Expression readPrefix()
```

zum Einlesen und Aufbau von Expressions. Die Expressions sollen in Prefix-Notation (Operator am Anfang) eingegeben werden, d.h. der Ausdruck $((2 * 3) + (6 / 7))$ wird eingegeben als

```
+ * 2 3 / 6 7
```

Hinweis:

Die Einleseoperation `readPrefix` ist rekursiv zu implementieren, d.h. wird z.B. ein `+` erkannt sind die Operanden von `+` mit der selben Methode `readPrefix` einzulesen.

Schreiben Sie dann einen benutzerfreundlichen Dialog zum Rechnen mit Expressions wie folgt:

Mit einem Zeichencode soll der Benutzer folgende Operationen auswählen können:

- R** - Einlesen eines Ausdrucks in Prefix-Notation
- +** - für eine Addition
- - für eine Subtraktion
- *** - für eine Multiplikation
- /** - für eine Division
- .** - für Ende

Jede Operation (außer `.`) liefert eine neue Expression als Ergebnis. Diese Expressions sollen in einem Array gespeichert werden und können für folgende Operationen als Operanden verwendet werden. Mit der Operation `R` wird eine neue Expression in Prefix-Notation eingegeben. Für die Operationen Addition, Subtraktion, Multiplikation und Division werden die Operanden durch Angabe des Arrayindex aus den Ergebnissen früherer Operationen ausgewählt und das Ergebnis im nächsten freien Arrayindex gespeichert. Nach jeder Operation werden die Expression und das Ergebnis seiner Berechnung ausgegeben.

Beispiel:

Expressions

=====

Folgende Operationen stehen zur Verfügung:

- R - Read: Einlesen einer neuen Expression
- + - Addition: Bilden einer Addition-Expression
- - Subtraction: Bilden einer Subtraction-Expression
- * - Multiplikation: Bilden einer Multiplication-Expression
- / - Division: Bilden einer Division-Expression
- . - Ende mit Punkt

```
-----
Kommando (R|+|-|*|/|.): r
Expression in Prefix-Notation: + 1 2
[0] (1 + 2) = 3
Kommando (R|+|-|*|/|.): r
Expression in Prefix-Notation: - 3 4
[1] (3 - 4) = -1
Kommando (R|+|-|*|/|.): -
Idx Operand 1: 0
Idx Operand 2: 1
[2] ((1 + 2) - (3 - 4)) = 4
Kommando (R|+|-|*|/|.): *
Idx Operand 1: 2
Idx Operand 2: 1
[3] (((1 + 2) - (3 - 4)) * (3 - 4)) = -4
Kommando (R|+|-|*|/|.):
```

Gehen Sie bei der Lösung folgend vor:

1. Implementieren Sie die Klassen und die Methoden in Java
2. Stellen Sie für alle Methoden einen Testplan auf und testen Sie das Programm.