Programmierung 2 (SS 2021)
Universität des Saarlandes
Fakultät MI
Compiler Design Lab

Musterlösung 12
Expressionsprojekt

Prof. Dr. Sebastian Hack
Julian Rosemann, B. Sc.
Thorsten Klößner, M. Sc.
Julia Wichlacz, M. Sc.
Maximilian Fickert, M. Sc.

Im Vorlesungskalender finden Sie Informationen über die Kapitel des Skripts, die parallel zur Vorlesung bearbeitet werden sollen bzw. dort besprochen werden. Die Übungsaufgaben dienen der Vertiefung des Wissens, das in der Vorlesung vermittelt wird und als Vorbereitung auf Minitests und Klausur.
Weitere Aufgaben zu den Themen finden Sie jeweils am Ende der Skriptkapitel.
Die Schwierigkeitsgrade sind durch Steine des 2048-Spiels gekennzeichnet, von 512 „leicht“ bis 2048 „schwer“. 4096 steht für Knobelaufgaben.

Aufgabe 12.0: Express Expressions

In dieser Aufgabe sollen Sie das von No Hau erzeugte unfertige Projekt *Expressions* vervollständigen, welches eine Repräsentation für einfache arithmetische Ausdrücke implementiert. Das dazugehörige Repository finden Sie unter `https://ProgII@gitlab.com/ProgII/Expressions.git`. Die genaue Spezifikation kann der Dokumentation entnommen werden. Gehen Sie zum Bearbeiten der Aufgabe wie folgt vor:

1. Erstellen Sie Klassen, die Ausdrücke mit den binären Operatoren $+, -, *, \div$ sowie das unäre Minus implementieren. Lassen Sie die Klassen dafür von den abstrakten Klassen `BinaryExpression` bzw. `UnaryExpression` erben. Ergänzen Sie Ihre Klassen zudem um passende Konstruktoren.
   Hinweis: Damit die Tests später reibungslos auf ihrer Implementierung laufen, nennen sie ihre Klassen nach dem folgenden Schema:AddExpression, SubExpression, ...
2. Implementieren Sie in den abstrakten Klassen `UnaryExpression`, `ConstExpression` und `BinaryExpression` die Methode `public String toString()`.
3. Implementieren Sie `public boolean equals(Object o)`. Achten Sie beim Vergleichen darauf, dass Ausdrücke, die bei Auswertung den gleichen Wert ergeben, sich gleichen sollen.
4. Vervollständigen Sie das Projekt, indem Sie die Methoden `public int eval()` implementieren und eine passende `Main`-Methode hinzufügen. Überprüfen Sie in dieser, ob Ihre Methoden korrekt funktionieren.

(Bonus) Erstellen Sie Klassen, die Ausdrücke mit den Operatoren $\wedge, \vee, \oplus$ und $\neg$ implementieren. Hierfür müssen Sie zusätzlich das `Enum Operators` erweitern. Für die Aufgabenteile 1, 2 und 4 gehen Sie vor, wie gehabt. Für Aufgabenteil 3 überlegen Sie sich, ob eine `equals()` Definition analog zu der Definition der arithmetischen Ausdrücke Sinn macht.

Lösung

Die Lösung der Aufgabe liegt auf dem Branch *deploy/basic/eval/solution* des angegebenen Repositories.
Die Lösung der Aufgabe mit Bonus liegt auf dem Branch *deploy/advanced/eval/solution* des angegebenen Repositories.