

Aufgabe 1

Parsen von Postfix-Notation:

Arithmetische Operationen lassen sich formal als Abbildung von einem Zahlenpaar auf das Ergebnis auffassen - so bildet etwa + das Paar (7, 8) auf die Zahl 15 ab. Nun kann man solche Operationen auf dreierlei Weise notieren:

Als

1. Präfix: +7 8,
2. Infix: 7 + 8 (uns wohl am Geläufigsten)
3. oder als Postfix: 7 8+

Wir wollen nun arithmetische Ausdrücke in Postfix-Notation auswerten und in der nächsten Aufgabe in Infix-Notation umwandeln. Der Vorteil der Postfix-Notation: wir benötigen keine Klammerung, wie bei der Infix-Notation (so entspricht $(2 + 3) * 4$ dem Ausdruck $23+4*,2 + (3 * 4)$ hingegen entspräche $234 * +$. Dadurch ist diese Notationsart leichter auszuwerten, und zwar folgendermaßen:

Wir durchlaufen die Zeichenkette, die auszuwerten ist, von links nach rechts. Wenn wir auf einen Operanden stoßen (eine Zahl also), legen wir sie auf den Stapel. Wenn wir auf einen Operator stoßen, holen wir die letzten zwei Operanden vom Stapel, verknüpfen diese dem Operator entsprechend und legen das Ergebnis auf den Stapel zurück. Wenn der String ganz durchlaufen wurde befindet sich genau ein Element auf dem Stapel: das Ergebnis.

Schreiben Sie eine Klasse Parser. Der Klassenmethode parse wird eine Zeichenkette in Postfixnotation übergeben und liefert deren Wert zurück.

Der Einfachheit halber lassen wir nur einstellige Operanden zu. Wir durchlaufen den String und betrachten immer genau ein Zeichen: die Methode char charAt(int i) der Klasse String liefert das i-te Zeichen. Die einhüllende Klasse Character besitzt die Methode int digit(char, int). Diese erlaubt die Transformierung in eine ganze Zahl bezüglich eines Zahlensystems.(Character.digit('7', 10) etwa liefert 7 als Integer. Ist das übergebene Zeichen nicht das einer Dezimalzahl, wird -1 geliefert).

In dem Ordner A des Ordners zur Vorlesung finden Sie die Klassen TestParser und KeinPostfixException. Verwenden Sie diese. Die Testklasse bezieht sich auch auf Aufgabe 2. Kommentieren Sie diesen Teil zunächst aus.

Letztgenannte Ausnahme soll geworfen werden, wenn beim Parsen etwas schief läuft. Es gibt drei Arten von Fehlern:

1. wir geraten beim Parsen an einen Operanden, aber auf dem Stapel befinden sich weniger als zwei Elemente
2. wir haben den String schon ganz durchlaufen, aber auf dem Stapel befindet sich mehr als ein Element
3. es kommen unerlaubte Zeichen vor - erlaubt sind 0, ...,9, +, -, * und /.

```
public class Parser {
    public double Parse(String Strng) {
        ArrayStack Stack = new ArrayStack();
        Double rslt = new Double(0f);
        Double op1 = new Double(0f);
        Double op2 = new Double(0f);
        for(int i = 0; i < Strng.length(); i++) {
            Character Char = new Character(Strng.charAt(i));
            //char c = Strng.charAt(i);
            System.out.println(Strng.substring(i));
            if(new String("+/*").indexOf(Char.toString()) >= 0) {
                op2 = (Double) Stack.pop();
                op1 = (Double) Stack.pop();
                if(Char.toString().indexOf("+") >= 0) {
                    rslt = new Double(op1.doubleValue() + op2.doubleValue());
                }
                else if(Char.toString().indexOf("-") >= 0) {
                    rslt = new Double(op1.doubleValue() - op2.doubleValue());
                }
                else if(Char.toString().indexOf("/") >= 0) {
                    rslt = new Double(op1.doubleValue() / op2.doubleValue());
                }
                else if(Char.toString().indexOf("*") >= 0) {
                    rslt = new Double(op1.doubleValue() * op2.doubleValue());
                }
                Stack.push(rslt);
            }
            else
                Stack.push(new Double(Double.parseDouble(Char.toString())));
            System.out.println(Stack.toString());
        }
        return rslt.doubleValue();
    }
}
```

```
public class TestParser {
    public static void main(String[] args) {
        Parser prsr = new Parser();
        double rslt = 0f;
        rslt = prsr.Parse("12+34*+2/22**");
        System.out.println("result: " + rslt);
    }
}
```

Aufgabe 2: Ergänzen Sie Parser um die Klassenmethode `String infix(String)`,

die nach Übergabe von zum Beispiel `13 + 2 * 2/3 - +` den String

`((((1 + 3) * 2)/2) + (7 - 3))`

zurückliefert, also Postfix in Infix transformiert. Das geht prinzipiell genauso, wie die letzte Aufgabe. Es werden allerdings keine ganzen Zahlen, sondern Strings auf den Stapel gelegt. Wenn wir auf einen Operator stoßen, wird nicht gerechnet, sondern drei Strings werden aneinandergehängt und in Klammern eingeschlossen.

Tipp: Um einen Character (`char`) `c` in einen String umzuwandeln, können Sie `""+c` verwenden.