

## Aufgabe 1: Syntaxanalyse

Nehmen Sie bei den folgenden Aufgaben an, dass Ihre Parserklasse eine Variable `scanner` besitzt, die der folgenden Schnittstelle gehorcht und dazu da ist, die Terminalsymbole des Eingabetextes zu erkennen.

```
enum Symbol { KL_AUF, KL_ZU, ... }

interface Scanner {
    Symbol lookahead();           // aktuelles Token
    void match(Symbol token);    // match-Aktion für Terminalsymbol
}
```

**1a)** Schreiben Sie eine Syntaxmethode für die folgenden Produktionsregeln (nur Analyse – keine Übersetzungaktionen):

```
ausdruck := term ( (PLUS | MINUS) term )* ;
```

(der Scanner wird über die Variable `s` angesprochen und `s.PLUS`, `s.MINUS` sind die Symbole, die der Scanner erkennt, das andere sind Nichtterminalsymbole)

**1b)** Was versteht man unter dem Builder-Muster. Erläutern Sie die Idee an einem kleinen Beispiel.

**1c)** Schreiben Sie eine Enum-Klasse für die Konstanten `BLUE`, `RED`, `GREEN`. Die Enum-Klasse soll eine Methode `getValue()` enthalten, die für `BLUE` (also `BLUE.getValue()`) den Wert 10, für `RED`, den Wert 77 und für `GREEN` den Wert -3 zurückgibt.

Die wichtigsten Details sollten erkennbar sein, syntaktische Kleinigkeiten sind nicht wichtig.

**1d)** Was ist die Aufgabe eines Scanners beim Einlesen einer Textdatei? Aufgabe 2: Nebenläufigkeit und Typparameter

Gegeben ist die folgende Klasse:

```
public class BoundedQueue {
    private List data = new ArrayList();

    public boolean isEmpty() { return data.isEmpty(); }
    public void put(Object x) { data.add(x); }
    public Object get() { return data.remove(0); }
}
```

## Aufgabe 2 Nebenläufigkeit

**2a)** Schreiben Sie die Klasse `BoundedQueue` so um, dass der Elementtyp durch einen Typparameter festgelegt wird.

**2b** Schreiben Sie die Methoden `put` und `get` so um, dass sie nicht nur threadsicher sind, sondern dass auch bei leerer Queue gewartet wird, bis die Operation ausgeführt werden kann.

**2c)** Die Schnittstelle `java.util.concurrent.Lock` enthält die folgenden Methoden

```
public void lock();
public void unlock();
```

Was muss man beachten wenn man einen kritischen Bereich mittels `lock/unlock` schützt (denken, Sie daran, was passiert, wenn evtl. in dem Bereich eine Exception geworfen wird). Schreiben Sie hin, wie der Schutz für `put` aus dem Queuebeispiel aussieht. (Es geht jetzt nicht um das Warten!)

**2d)** Was ist die Besonderheit bei einem `ReadWriteLock`? **2e)** Schreiben Sie die folgende for-Schleife so um, dass sie durch einen älteren Java-Compiler (vor Java 1.5) übersetzt werden kann:

```
ArrayList<Double> liste = ...
double summe = 0;
for (double x : liste) s += x;
```

### Aufgabe 3: Verteilte Systeme

**3a)** Führen Sie vollständig auf, welche Datentypen als Parametertypen und als Rückgabetyper eines RMI-Aufrufs auftreten können. Worin unterscheidet sich das Verhalten der Parameterübergabe im Einzelnen.

**3b)** Führen Sie möglichst vollständig die Aufgaben eines Stubs im Rahmen der RMI-Kommunikation auf.

**3c)** Erläutern Sie die Begriffe “*sequentieller Server*”, “*paralleler Server*” und “*Threadpool*”.

**3d)** Welche unterschiedlichen Aufgaben nehmen die Klassen `ServerSocket` und `Socket` wahr?