Aufgabe 8: Übersetzung von XML nach Json

Sowohl XML (*Extensible Markup Language*) als auch Json (*JavaScript Object Notation*) sind Formate zur Beschreibung markierter Datensätze. Ein XML-Datensatz, bestehend aus mehreren XML-Elementen,

```
<person>
  <name>Meier</name>
  <city>Wien</city>
</person>
<person>
  <name>Huber</name>
  <city>Berlin</city>
</person>
```

wird in Json wie folgt dargestellt:

```
[
  person: {
    name: Meier,
    city: Wien
  },
  person: {
    name: Huber,
    city: Berlin
  }
]
```

Wir verwenden in diesem Beispiel eine vereinfachte XML-Syntax:

```
XML     = { Element } .
Element = "<" ident ">"
            (  ident
            |  { Element }
            )
            "</" ident ">".
```

Attributieren Sie diese Grammatik, sodass ein XML-Datensatz in einen Json-Datensatz übersetzt wird. Prüfen Sie auch, dass die Tag-Namen in <ident> und </ident> übereinstimmen.

Lösung

Die Attributierung ist im Prinzip einfach. Wir holen uns den Tag-Namen (z.B. den Namen des Tags <city>) sowie den Inhalt eines XML-Elements (z.B. Wien) und generieren daraus das Json-Format. Eine kleine Herausforderung sind die korrekten Einrückungen sowie das richtige Setzen von Beistrichen in Json-Listen.Für die Einrückungen geben wir jedem Element als Attribut mit, wie weit es eingerückt werden soll. Beim Setzen der Beistriche in Element-Listen achten wir darauf, ob ein Element das erste der Liste ist (in diesem Fall müssen wir davor keinen Beistrich setzen) oder ob es ein Folgeelement in der Liste ist (in diesem Fall setzen wir davor einen Beistrich).

```
XML
=                          (. boolean first = true;
                              System.out.print("["); .)
 { Element <↓1, ↓first>    (. first = false; .)
 }                         (. System.out.println();
                              System.out.println("]");.) .
```

Die Elemente auf äußerster Ebene werden um 0 Zeichen eingerückt. Beim ersten Element ist first = true, bei allen anderen ist first = false. Die Produktion von Element sieht wie folgt aus:

```
Element <↓indent, ↓first>          (. int indent; boolean first; String tag, val, s; .)
= "<" Ident <↑tag> ">"             (. if (first) System.out.println(); else System.out.println(","); .)
   (  Ident <↑val>                 (. Indent(indent);
                                      System.out.print(tag + ": " + val); .)
   |                               (. Indent(indent);
                                      System.out.print(tag + ": {");
                                      first = true; .)
      { Element <↓indent+2, ↓first> (. first = false; .)
      }                            (. System.out.println();
                                      Indent(indent);
                                      System.out.print("}"); .)
   )
   "</" Ident <↑s> ">"             (. if (! s.equals(tag)) error("non-matching tag"); .)  .
```

Wenn ein Element eine geschachtelte Element-Liste enthält, rücken wir diese um 2 Zeichen weiter ein als die umgebende Liste. Für das erste Element dieser Liste ist first wieder true, für die weiteren Elemente ist es false. Um Einrückungen auszugeben, verwenden wir eine Methode Indent:

```
void Indent (int n) {
   for (int i = 0; i < n; i++) System.out.print(" ");
}
```

Für Namen deklarieren wir ein Nonterminalsymbol Ident, das das Terminalsymbol ident erkennt und dessen Wert (t.val) als Attribut zurückgibt:

```
Ident <↑s>          (. String s; .)
= ident             (. s = t.val; .) .
```

Dies ist eine gängige Technik, um Terminalsymbole mit Attributen zu versehen.