Aufgabe 3: Einlesen eines Telefonbuches

Personen und ihre Telefonnummern seien wie folgt in einer Textdatei gespeichert:

```
Mustermann, Karl Otto
    home +49 (6221) 24684567,
    work (0151) 234567
Meier, Anna
    23456789
...
```

Jeder Eintrag beginnt mit einem Nachnamen, einem Komma und einem oder mehreren Vornamen. Darauf folgt eine durch Kommas getrennte Liste von Telefonnummern, die eine der folgenden Formen haben kann:

+49 (6221) 24684567	Ländercode, Vorwahl in Klammern ohne 0, Nummer
(0151) 234567	Vorwahl in Klammern, Nummer
23456789	Nummer ohne Ländercode und Vorwahl

Vor jeder Telefonnummer kann die Angabe "home" oder "work" stehen. Wenn sie fehlt, wird "home" angenommen. Wenn der Ländercode fehlt, wird "+49" angenommen, wenn die Vorwahl fehlt, wird "030" angenommen.

a) Beschreiben Sie das Format dieser Textdatei durch eine kontextfreie Grammatik.

b) Attributieren Sie diese Grammatik, sodass die Namen und Telefonnummern gelesen und in einer geeigneten Datenstruktur gespeichert werden. Geben Sie diese Datenstruktur zur Kontrolle aus. Erzeugen Sie mittels Coco/R ein Programm, das die Eingabedatei liest, daraus die besagte Datenstruktur aufbaut und diese zur Kontrolle ausgibt.

Lösung

a) *Kontextfreie Grammatik*

```
PhoneBook   = Entry {Entry}.
Entry       = Name Number {"," Number}.          // a person can have multiple phone numbers
Name        = ident "," ident {ident}.           // last name, first name(s)
Number      = [Kind] [CountryCity | City] number.
Kind        = "home" | "work".
CountryCity = "+" number "(" number ")".         // country code and city code without leading 0
City        = "(" number ")".                    // city code with leading 0
```

b) *Compilerbeschreibung*

```
import java.util.ArrayList;

COMPILER PhoneBook

  // ------------------ global declarations --------------------

  class Person {
    String firstName;
    String lastName;
    ArrayList<Phone> numbers;
    Person(String first, String last) {
      firstName = first; lastName = last;
      numbers = new ArrayList<Phone>();
    }
  }
```

```
   class Phone {
      String kind;                  // "home", "work"
      String countryCode;           // e.g., "+49"
      String cityCode;              // e.g., "06221"
      String number;                // e.g., 123456
      Phone (String k, String co, String ci, String n) {
         kind = k; countryCode = co; cityCode = ci; number = n;
      }
   }

   void printPhoneBook() {
      for (Person p: phoneBook) {
         System.out.println(p.lastName + ", " + p.firstName);
         for (Phone phone: p.numbers) {
            System.out.println("  " + phone.kind + " " + phone.countryCode + " " + phone.cityCode + " " + phone.number);
         }
      }
   }

   ArrayList<Person> phoneBook = new ArrayList<Person>();

//------------------- scanner specification ------------------
CHARACTERS
  letter    = 'A'..'Z' + 'a'..'z'.
  digit     = '0' ..'9'.

TOKENS
  ident = letter {letter}.
  number = digit {digit}.

IGNORE '\r' + '\n' + '\t'

//------------------- parser specification ------------------
PRODUCTIONS

PhoneBook = Entry { Entry }     (. printPhoneBook(); .) .

Entry                           (. Person person; Phone phone; .)
= Name <out person>
  Number <out phone>            (. person.numbers.add(phone); .)
  {    ","
       Number <out phone>       (. person.numbers.add(phone); .)
  }                             (. phoneBook.add(person); .) .

Name <out Person person>        (. String firstName, lastName; .)
=  ident                        (. lastName = t.val; .)
   "," ident                    (. firstName = t.val; .)
   {  ident                     (. firstName += " " + t.val; .)
   }                            (. person = new Person(firstName, lastName); .) .

Number <out Phone phone>        (.  String kind = "home";       // default
                                    String countryCode = "+49"; // default
                                    String cityCode = "030";    // default
                                    String number; .)
=  [ "home"                     (. kind = "home"; .)
   | "work"                     (. kind = "work"; .)
   ]
   [   "+" number               (. countryCode = "+" + t.val; .)
       "(" number               (. cityCode = "0" + t.val; .)
       ")"
   |   "(" number               (. cityCode = t.val; .)
       ")"
   ]
   number                       (. phone = new Phone(kind, countryCode, cityCode, t.val); .) .

END PhoneBook.
```

Als Terminalsymbole haben wir (neben den Literalen) lediglich Namen (ident) und Zahlen (number). Neben Leerzeichen sollen Zeilenenden und Tabulatoren ignoriert werden.

Als Datenstruktur für das Telefonbuch verwenden wir eine Liste phoneBook, die in den globalen semantischen Deklarationen deklariert ist. Die Liste enthält Personen (Klasse Person mit Vorname, Nachname und Telefonnummern). Die Telefonnummern sind ebenfalls eine Liste von Phone-Objekten mit der Art des Eintrags (home oder work), dem Ländercode, der Vorwahl und der eigentlichen Telefonnummer.

Ein Eintrag besteht aus einem Namen (Name) und einer Folge von Telefonnummern (Number). Name liefert als Ausgangsattribut ein Person-Objekt, Number liefert ein Phone-Objekt. Diese Objekte werden in die Datenstruktur eingetragen.

Die Methode printPhoneBook gibt die Datenstruktur phoneBook zur Kontrolle aus.

Wir brauchen noch ein Hauptprogramm, das den Namen der Eingabedatei als Kommandozeilenparameter liest, einen Scanner und einen Parser erzeugt und die Syntaxanalyse startet:

```
class PhoneBook {
   public static void main (String[] arg) {
      Scanner scanner = new Scanner(arg[0]);
      Parser parser = new Parser(scanner);
      parser.Parse();
      System.out.println(parser.errors.count + " errors detected");
   }
}
```

Wir verarbeiten nun die Compilerbeschreibung mit Coco/R:

```
java -jar Coco.jar PhoneBook.atg
```

wodurch ein Scanner und ein Parser erzeugt wird. Anschließend übersetzen wir alle Java-Dateien

```
javac Scanner.java Parser.java PhoneBook.java
```

und führen dann den Interpreter wie folgt aus:

```
java PhoneBook input.txt
```

Die Datei input.txt enthält die Namen und Telefonnummern wie in der Aufgabenstellung beschrieben. Die Ausgabe lautet für das oben angegebene Beispiel:

```
Mustermann, Karl Otto
   home +49 06221 24684567
   work +49 0151 234567
Meier, Anna
   home +49 030 23456789
...
```