Aufgabe 1: Pfadberechnung

Gegeben sei eine Textdatei mit einer Folge von Punkten im Format (n, m), wobei n und m ganzzahlige Konstanten sind. Stellen Sie eine Grammatik dieser Punktfolge auf und schreiben Sie mittels Coco/R ein Programm, das die Länge des Pfades vom ersten bis zum letzten Punkt berechnet und ausgibt.

Lösung

Die kontextfreie Grammatik der Eingabedatei ist einfach. Sie lautet:

```
Path = Point {Point}.
Point = "(" number "," number ")".
```

Bei der Attributierung erzeugen wir für jeden Point ein Punkt-Objekt mit x- und y-Koordinate und berechnen die Entfernung zwischen zwei Punkten nach Pythagoras.

```
COMPILER Path

//---------------------- global declarations ----------------------
class Pt { int x, y; }  // class for point objects

//---------------------- scanner specification ----------------------
CHARACTERS
  digit   = '0'..'9'.

TOKENS
  number    = digit {digit}.

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

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

Path                (.  Pt p, q; .)
=  Point <out p>    (.  double length = 0; .)
   { Point <out q>  (.  int dx = q.x - p.x;
                        int dy = q.y - p.y;
                        length += Math.sqrt(dx*dx + dy*dy);
                        p = q; .)
   }                (.  System.out.println("path length = " + length); .) .

Point <out Pt p>
=  "(" number       (.  p = new Pt();
                        p.x = Integer.parseInt(t.val); .)
   "," number       (.  p.y = Integer.parseInt(t.val); .)
   ")" .

END Path.
```

In der Scannerbeschreibung deklarieren wir das Terminalsymbol number und legen fest, dass neben Leerzeichen auch Zeilenenden und Tabulatoren ignoriert werden sollen.

Die Parserbeschreibung enthält die attributierte Grammatik, die ohne weitere Erklärungen verständlich sein sollte.

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

```
class Path {
   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 Path.atg
```

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

```
javac Scanner.java Parser.java Path.java
```

und führen dann das Programm wie folgt aus:

```
java Path input.txt
```

Die Datei input.txt enthält eine Folge von Punkten und könnte wie folgt aussehen:

```
(0, 0) (2, 1) (3, 3) (5, 3) (6, 1)
```

Die Ausgabe ist in diesem Fall

```
8.70820393249937
```