Exercise 1: Path Calculation

Given a text file containing a sequence of points with coordinates in the format (n, m), where n and m are integer constants. Write an attributed grammar that processes this input by calculating and printing the length of the path from the first to the last point. Use Coco/R to generate a scanner and a parser and write a main program that starts the processing.

Solution

The context-free grammar of the input is simple:

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

When turning this into an attributed grammar we generate from each Point an object of class Pt with an x and y coordinate and compute the distance between two points according to 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 the scanner specification, we declare the terminal symbol number and specify that, in addition to blanks, line endings and tabs should be ignored.

The parser specification contains the attributed grammar, which should be understandable without further explanation.

We still need a main program that gets the name of the input file as a command line argument, generates a scanner and a parser and starts the syntax analysis:

```
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");
   }
}
```

We process the compiler specification with Coco/R:

```
java -jar Coco.jar Path.atg
```

thus generating a scanner and a parser. Then we compile all Java files

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

and invoke the program as follows:

```
java Path input.txt
```

Assuming that the file input.txt contains the following sequence of points

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

the output is

```
8.70820393249937
```