```java
import org.antlr.v4.runtime.ANTLRInputStream;
import org.antlr.v4.runtime.CommonTokenStream;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MyLanguageRunner {

    public static void main(String[] args) throws Exception {

        // check if we want to use a file as input or System.in
        String inputFile = null;
        if (args.length > 0) inputFile = args[0];
        InputStream is = System.in;
        if (inputFile != null) {
            is = new FileInputStream(inputFile);
        }

        BufferedReader br = new BufferedReader(new InputStreamReader(is));
        String expr = br.readLine(); // get first line of input
        int line = 1; // track line numbers




        // create a Parser that we will reuse for each line of input
        // ** change name of Parser to match your Parser name

        // we will share this single parser instance with different lexers

        MyLanguageParser parser = new MyLanguageParser(null);
        parser.setBuildParseTree(false); // don't need trees

        // as long as we keep getting input we create a new LEXER that will
        // generate a new set of TOKENS to feed to our parser.

        while (expr != null) { // while we have more lines of input
            // create new lexer and token stream for each line (expression)
            ANTLRInputStream input = new ANTLRInputStream(expr + "\n");

            // ** change name of Lexer to match your Lexer
            MyLanguageLexer lexer = new MyLanguageLexer(input);

```

```java
                // do some lexer work
                lexer.setLine(line); // notify lexer of input position
                lexer.setCharPositionInLine(0);
                CommonTokenStream tokens = new CommonTokenStream(lexer);

                // pass our TOKENS to the parser
                parser.setInputStream(tokens); // notify parser of new token stream

                // ** change 's' to your starting parser rule
                parser.root_rule(); // start the parser to match rule s

                expr = br.readLine(); // see if there's another line
                line++;
            }
        }
}
```