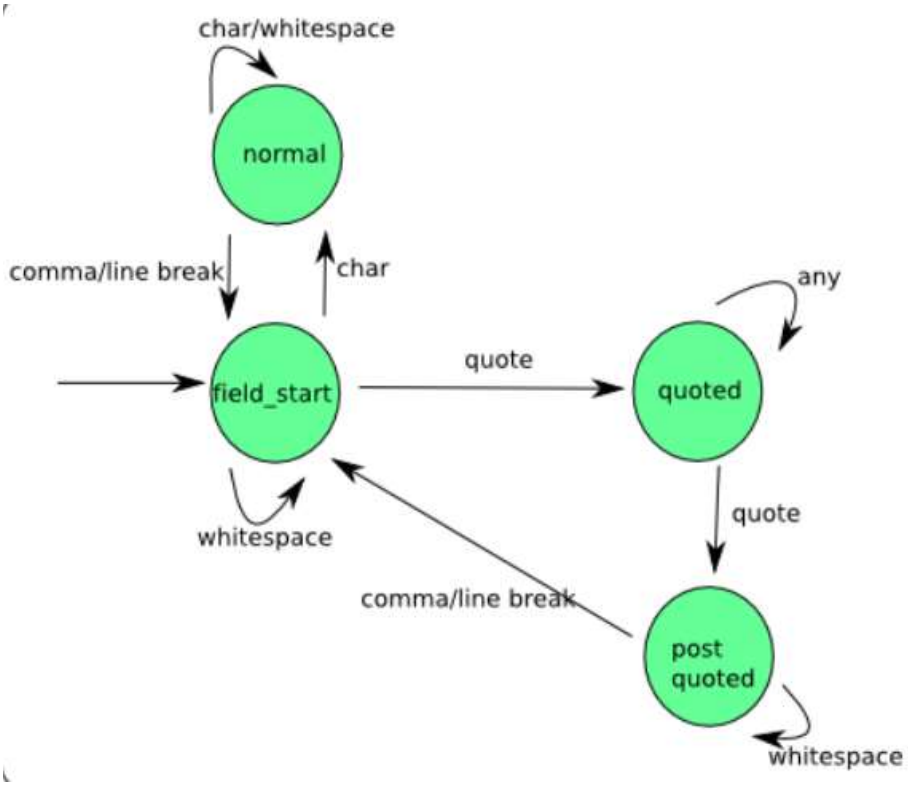
[A12] Zustandsautomat

Grafische Beschreibung



Beispiel eines CSV Parsers

Textuelle Beschreibung

Ein Zustandsautomat, auch endlicher Automat genannt, wird zur Modellierung eines Verhaltens genutzt und verfügt dabei über Zustände, Zustandsübergangen und Aktionen. Je nachdem welchen Zustand ein Automat gerade besitzt, werden bei Input/Eingabe die jeweiligen Zustandsübergänge genutzt.

In diesem Fall wurde ein Zustandsautomat, welcher ein CSV File einliest und als String verarbeitet, entwickelt. Hierbei werden Texte und Zeichen abgefragt, aber auch Fehler abgefangen. Der Automat verfügt dabei über Stati wie Text, Comma, Quote, CR, LF, TAB und Error. Die einzelnen Übergänge bauen den String auf.

Laufzeit

- **O(n),** da jedes Element aus dem CSV durchlaufen wird.

Wichtige Elemente

Der Code lässt sich auch auf unserem Github übersichtlicher einsehen.

```
public static CSVResult parse(String str) {

   System.out.println("Input String: " + str + " length: " + str.length());

   CSVResult result = new CSVResult();
   States state;

   boolean quoteHelper = false;

   int countQuote = 0;
   int countLf = 0;
   int countCr = 0;
   int lastChar = -1;

   parseLoop: for (int i = 0; i < str.length(); i++) {

      char c = str.charAt(i);
      state = getStateLoop(result, c);

      switch(state) {

         case ERROR:       // ++++++++++++++++++++++++++ ERROR ++++++++++++++++++++++++++++++
            break parseLoop;

         case TEXT:     // ++++++++++++++++++++++++++ TEXT ++++++++++++++++++++++++++++++
            result.appendChar(c);
            break;

         case COMMA:       // ++++++++++++++++++++++++++ COMMA ++++++++++++++++++++++++++++++
            result.addValue();
            break;

         case QUOTE:       // ++++++++++++++++++++++++++ QUOTE ++++++++++++++++++++++++++++++
            countQuote++;

            if (lastChar != '\"') quoteHelper = false; // check if last char is not a QUOTE

            if (quoteHelper) {
               if ((i != str.length() - 1) && (str.charAt(i + 1) != ','))  // check if NOT
last element in string AND next element is NOT a COMMA
                  result.appendChar(c);
               quoteHelper = false;
            }
            else
               quoteHelper = true;

            break;

         case CR:      // ++++++++++++++++++++++++++++++ CR ++++++++++++++++++++++++++++++
            countCr++;
            if (!(str.length()-2 == i))    // check if carriage return is NOT 2nd to last
element in string
               result.appendChar(c);
            break;

         case LF:      // ++++++++++++++++++++++++++++++ LF ++++++++++++++++++++++++++++++
            countLf++;
            if (!(str.length()-1 == i))    // check if line feed is NOT last element in string
               result.appendChar(c);
            break;

         case TAB:     // ++++++++++++++++++++++++++++++ TAB ++++++++++++++++++++++++++++++
            result = CSVResult.ERROR;  // always error if TAB within string (?)
            break;

         default:
            System.out.println("Input could not be parsed - set to ERROR");
            result = CSVResult.ERROR;
      }

      lastChar = c;

   }

   // check if string parsing was successful based on result state and counter variables
   state = getStatePost(result, countCr, countLf, countQuote);

   switch(state) {
```

```
            case ERROR:
                result = CSVResult.ERROR;
                break;

            case SUCCESSFUL:
                result.addValue(); // add last element to array list
                break;

            default:
                System.out.println("Result could be evaluated - set to ERROR");
                result = CSVResult.ERROR;
        }

        return result;
    }
```