SNLP Assignment 1

Clemens Damke - 7011488

1. Regular Expressions

Since it wasn't specified what exactly a word is, I assumed that `[A-Za-z]` describes valid word characters.
Additionally I assumed that by *"the"*, *"theo"* and *"a"* only strings with that capitalization were meant.

1. ***"the"* as a real substring:** `the[A-Za-z]+|[A-Za-z]+the[A-Za-z]*`
2. ***"the"* but not *"theo"*:** `the[^o][A-Za-z]*|[A-Za-z]+the([^o][A-Za-z]*|)`
3. ***"a"* at least 3 times:** `[A-Za-z]*a[A-Za-z]*a[A-Za-z]*a[A-Za-z]*`

2. Finite-State Automata

2.1. D-RECOGINZE

`task21.jar` contains the solution to the first part of this task.
Source code is included in the jar.
As required it has to be called with three arguments: `java -jar task21.jar transitionPath inputPath outputPath`.
All three have to paths.

- `transitionPath`: Path to a file that contains a transition table
- `inputPath`: Path to a file that contains the input for the DFA.
- `outputPath`: Path to a file to which the output will be written.

Since there were no requirements on the accepted input alphabet, the implementation only supports ASCII to keep the code simpler.

2.2. Every match

`task22.jar` contains the solution to the second part of this task.
It has to be called with two arguments: `java -jar task21.jar transitionPath inputPath`.
Output will be written to stdout so there is no `outputPath` option.
The implementation is pretty inefficient since it simply iterates over all suffixes of the input and

matches against them.
Could be improved by caching the results of `(state, input position)` pairs over all iterations, determining states that always reject to terminate early if possible and maybe add multithreading to match multiple suffixes in parallel...
Optimizations not implemented because: Too lazy and not required.