1 Notation

The grammar for the Lando System Specification Sublanguage is written in the
EBNF notation. The main elements of the notation that we utilize are:

- Terminals are represented with double or single quotes; e.g. `"explanation"` .

- Optional bits are represented with squared brackets; e.g. [`"explanation"` paragraph].

- Repetition is represented with curly braces; e.g. { identifier }

- We use a slightly enhanced notation - { a }$^{+}$ to indicate *non-zero* repetitions.
  This is simply equivalent to: a { a }.

- For defining terminals, we would like to use the EBNF special form to declare
  an extended regular expression - for example: ?$/\w+/$?. However since this
  is rather verbose, we will simply use $/\w+/$ for convenience.

- Blocks in the language are typically delimited by keywords indicating the start
  of *another* block. In defining the grammar this translates to lookaheads:
  (i.e. peeking at incoming tokens without consuming them). We use the perl
  regular expression format to indicate this. E.g. $/?=$ (new-line `"system"`)$/$.

2 Grammar

lando-source	::=	{ spec-element }	*Lando source*
spec-element	::=	system | subsystem | component | event	
		| scenario | requirement	*Specification Elements*
system	::=	`"system"` name-phrase-rel [rel-keyword name-phrase] new-line	
		explanation new-line	
		[`"indexing"` new-line indexing new-line]	
		subsystem { new-line subsystem }*	
		$/(?=$ nl-sys-keyword | eof $)/$ block-end	*System*
subsystem	::=	`"subsystem"` name-phrase-rel [rel-keyword name-phrase] new-line	
		explanation new-line	
		[`"indexing"` new-line indexing new-line]	
		component { new-line component }*	
		$/(?=$ nl-subsys-keyword | eof $)/$ block-end	*Cluster*
component	::=	`"component"` name-phrase-rel [rel-keyword name-phrase] new-line	
		component-part { new-line component-part }*	
		$/(?=$ nl-keyword | eof $)/$ block-end	*Class*
component-part	::=	constraint | command | query	*Component Parts*
constraint	::=	$/[\hat{}.?!]^{+}?\backslash.]/m$	*Constraint*
query	::=	$/[\hat{}.?!]^{+}?\backslash?]/m$	*Query*
command	::=	$/[\hat{}.?!]^{+}?!]/m$	*Command*

event	::=	`"events"` name-phrase new-line	
		event-entry { new-line event-entry }*	
		/(?= nl-keyword | eof)/ block-end	*Events*
event-entry	::=	identifier new-line sentence	*Event Entry*
scenario	::=	`"scenarios"` name-phrase new-line	
		event-entry { new-line event-entry }*	
		/(?= nl-keyword | eof)/ block-end	*Scenario*
scenario-entry	::=	identifier new-line sentence	*Scenario Entry*
requirement	::=	`"requirements"` name-phrase new-line	
		req-entry { new-line req-entry }*	
		/(?= nl-keyword | eof)/ block-end	*Requirements*
req-entry	::=	identifier new-line sentence	*Requirements Entry*
indexing	::=	index-entry { new-line index-entry }*	*Index List*
index-entry	::=	index-key `':'` index-val-list	*Index List*
index-key	::=	/[^:]$^+$/	*Index Key*
index-val-list	::=	index-val { new-line index-val }*	
		/(?= eof | nl-keyword | new-line index-key)/	*Index Value List*
index-val	::=	/[^:]$^+$/	*Index Value*
name	::=	identifier	*Name*
name-phrase-rel	::=	/\w[\w\s]* (?= rel-keyword | new-line)/	*Name-Phrase*
sentence	::=	/[^.?!]$^+$? [.?!]/m	*Sentence*
paragraph	::=	sentence $^+$/(?= (new-line keyword | eof))/	*Paragraph*
explanation	::=	paragraph	*Explanation*
keyword	::=	< *allkeywords* >	*All Keywords*
nl-keyword	::=	new-line keyword	*Keyword on new line*
nl-sys-keyword	::=	new-line `"system"`	
nl-subsys-keyword	::=	new-line (`"subsystem"` | `"system"`)	
rel-keyword	::=	`"inherit"` | `"client"` | `"contains"`	*Relation keywords*
block-end	::=	new-line | eof	*Block End*
identifier	::=	/\w+/	*Identifier*
new-line	::=		*New Line*