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.

2 Grammar

lando-source	::=	{ spec-element }	(*Lando source*)
spec-element	::=	informal-chart | class-dictionary	(*Specification Elements*)
informal-chart	::=	subsystem | cluster-chart | class-chart | event-chart | scenario-chart | requirements-chart | creation-chart	(*Informal Charts*)
class-dictionary	::=	`"dictionary"` name $\{$ dictionary-entry $\}^+$ `"end"`	(*Class Dictionary*)
dictionary-entry	::=	`"class"` name `"description"` paragraph	(*Class Dictionary Entry*)

subsystem	::=	`"system_chart"` name new-line explanation [`"indexing"` indexing new-line] { cluster-entry } `"end"`	(*System Chart*)
cluster-entry	::=	`"cluster"` name new-line explanation	(*Cluster Entry*)

cluster-chart	::=	`"cluster_chart"` name
		[`"explanation"` paragraph]
		{ class-entry }
		{ cluster-entry }
		`"end"`
			(*Cluster Chart*)

class-entry	::=	`"class"` name `"description"` paragraph	(*Class Entry*)

class-chart	::=	`"class_chart"` name [`"explanation"` paragraph] [`"inherit"` name-list] [`"query"` string-list]	

```
                            [ "command" string-list ]
                            [ "constraint" string-list ]
                            "end"                                        (Class Chart)
------------------------------------------------------------------------------------------
     event-chart    ::=     "event_chart" name
                            [ event-direction ]
                            [ "explanation" paragraph ]
                            { event-entry }
                            "end"                                        (Class Chart)
     event-entry    ::=     "event" sentence "involves" name-list        (Event Entry)
 event-direction    ::=     "incoming" | "outgoing"                      (Event Direction)
------------------------------------------------------------------------------------------
  scenario-chart    ::=     "scenario_chart" name
                            [ "explanation" paragraph ]
                            { scenario-entry }
                            "end"                                        (Scenario Chart)
  scenario-entry    ::=     "scenario" name "description" paragraph      (Scenario Entry)
------------------------------------------------------------------------------------------
  creation-chart    ::=     "creation_chart" name
                            [ "explanation" paragraph ]
                            { creation-entry }
                            "end"                                        (creation Chart)
  creation-entry    ::=     "creator" name "creates" name-list           (creation Entry)
------------------------------------------------------------------------------------------
        indexing    ::=     { index-entry }                              (Index List)
     index-entry    ::=     identifier ':' string-list new-line          (Index List)
------------------------------------------------------------------------------------------
            name    ::=     identifier                                   (Name)
       name-list    ::=     name { ',' name }                            (Name List)
          string    ::=     /[^,]+?/                                     (String)
     string-list    ::=     string {, string }                           (List of Strings)
        sentence    ::=     /[^.?!]+? [.?!]/m                            (Sentence)
   sentence-list    ::=     sentence { ',' sentence }                    (String List)
       paragraph    ::=     sentence +/(?= (new-line keyword | eof ))/   (Paragraph)
     explanation    ::=     paragraph                                    (Explanation)
         keyword    ::=     < allkeywords >                              (All Keywords)
      identifier    ::=                                                  (Identifier)
        new-line    ::=                                                  (New Line)
```