Rationale of the RAQUEL Token Class Physical Design

The constituents of Raquel Tokens vary considerably, and depend on what an individual token represents. The design of an object class to hold Raquel Tokens would traditionally exploit class inheritance in order to cope with such variability. The PoC prototype does this.

However the use in the prototype of many object classes linked together in an inheritance hierarchy created great complexity. That is contrary to the design aim of ***maximum simplicity***. To achieve this, the design should attempt to follow principles explained in the paper "*Out of the Tar Pit*" written by Ben Moseley and Peter Marks :

1. Avoid complexity wherever possible.
2. Separate out components wherever possible.
3. Do not prematurely design for performance.[1]

Hence the re-factored design has the following properties :

1. It uses ***one*** object class design for ***all*** Raquel Tokens, regardless of what a token represents. No class inheritance hierarchy is used.

   Consequently parts of a token may be unused in any particular instance. Since only a few bytes are wasted in even the most extreme case, this is considered worthwhile.

   What a token object represents is determined by the contents of its data members.

2. The chief variability in size and nature of token contents stems from the parse trees used to represent parameters of operators and assignments. There are 2 possible types of parameter parse tree, representing :
   - expressions that are parameters,
   - literal relational/container values that are parameters.

   Both kinds of parameter are logically part of a Raquel Token.[2] However the nodes of their parse trees are also Raquel Tokens. Since it is impractical to physically hold tokens within tokens, the parameter parse trees are physically separate from the Raquel Token of which they are logically a part; but physically accessible only from that token. Hence they are *de facto* encapsulated within the token object, even if not literally so.

   Other parameter data that can vary in size is held in the form of lists of names, and is held literally within the token. Its size variation is much less than that of parse trees.

---

[1] After system completion, any ***actual*** performance bottlenecks should be found by monitoring performance, and only those re-designed to give improved performance.

[2] A Raquel Token's parameter parse trees should be distinguished from that parse tree in which the Raquel Token is itself a node.