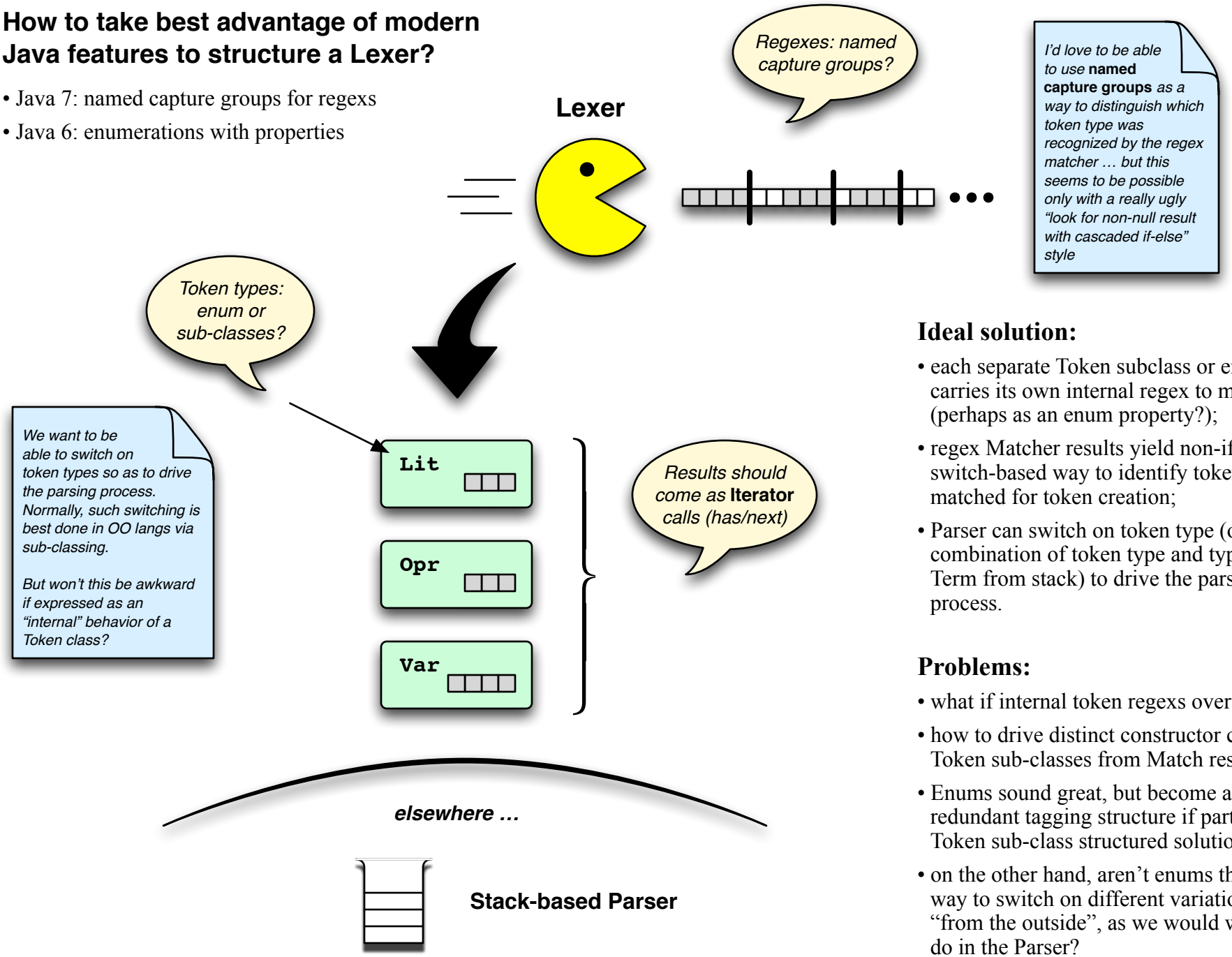
How to take best advantage of modern Java features to structure a Lexer?

- Java 7: named capture groups for regexs
- Java 6: enumerations with properties



Ideal solution:

- each separate Token subclass or enum carries its own internal regex to match on (perhaps as an enum property?);
- regex Matcher results yield non-if/ switch-based way to identify token type matched for token creation;
- Parser can switch on token type (or combination of token type and type of Term from stack) to drive the parsing process.

Problems:

- what if internal token regexs overlap?
- how to drive distinct constructor calls for Token sub-classes from Match results?
- Enums sound great, but become a redundant tagging structure if part of a Token sub-class structured solution.
- on the other hand, aren't enums the best way to switch on different variations "from the outside", as we would wish to do in the Parser?