Summary of Parsers in Smalltalk" class. Thursday 2006.5.11

- View Student's code.
  - different lengths for methods to build DFSM from NFSM
  - note that it's OK to use a Set as a key to a dictionary
- Streams
  - compare with iterators
    - Smalltalk has internal (do: family) and external iterators (streams)
  - simple stream usage
    - change printOn: for Streams to show current position
      - useful technique while debugging
  - program mergesort
    - example of Read- and WriteStreams
- Streams as a basis for parsers
  - Streams keep track of the text that we are parsing
- Parsers and Blocks
  - Compositional programming
  - Parser combinators considered as program-building operations
    - Make sure that the program that you are building is not infinite!
- class comment from ParserStream (this class was renamed; it was APParser in class)
  - This class is used to generate composable parsers, patterned after Haskell parser combinators.

A parser is actually a block. The result of evaluating the block is:
  . if the parse succeeds: a sequence of correctly parsed items
  . if the parse fails: nil.

The effect of evaluating the block is to advance the underlying stream if the parse succeeds, and not to advance it if the parse fails. This invariant is maintained across compound parsers.

- Refactoring the parsers
  - alternatives to the value ifNil: [nil] ifNotNil: [ <the rest of the parser…> ] convention: