EXTERNAL STORAGE & DataRecords

- For a FILE, individual records are each written to the file "**immediately"** after their data is collected and transformed/cleaned. *[Actually RawData class reads/transforms/cleans the data – DataTable/DataRecord classes do the actual constructing of the dataRecord (including truncating/padding it to make it fixed-length].* All records belonging in the file are **NOT** :
    - FIRST constructed and temporarily stored in an internal storage structure (e.g., in an array or arraylist),
    - then at the EOF on the input, writing the array to the file.

  Setup uses the classic "Input Stream Processing" design pattern, as was done in A1.
  That is, after RawData gets a record (since Setup asked for it)
  Setup calls dataTable.Insert with the 7 individual fields as parameters
  (rawData.getCode, rawData.getId, . . . rawData.getLifeExp)
  which in turn invokes dataRecord.Write1Record(int RRN)

- Since the file uses Direct Address, records are written to the file IN THE APPROPRIATE LOCATION when they're written to the file. There is NO SORTING involved since Direct Address results in the file being "naturally sorted" on the D.A. key.

- Similarly during processing (DI, SI, AI), only a **SINGLE record** is read at a time, with subsequent records read (e.g., on a future DI/SI/AK into the SAME, SINGLE record-storage location (object).
    - So there's only ever a SINGLE record object allocated for storage internally. [There's only **1 object of the dataRecord class** declared].

- **Implementation** issue: DataRecord class has these 4 methods:
    - Write1Record(int RRN) – for random access writing which
        1. Calculates the byteOffset
        2. Does a seek
        3. Calls Write1Record()
    - Read1Record(int RRN) – for random access reading which
        1. Calculates the byteOffset
        2. Does a seek
        3. Calls Read1Record()
    - Read1Record() – for sequential access AND for reading whatever record is at the file-position pointer (set by Read1Record(int RRN) method above)
    - Write1Record() - for sequential access writing (which is NEVER DONE in this project) AND for writing whatever record is to go at the file-position pointer (set by Write1Record(int RRN) method above)

- The implementation (body) of Read1Record and Write1Record may/will be different for different programmers, depending on:
    - what library methods you use
    - whether you're reading/writing a LINE or a RECORD or FIELD-BY-FIELD
    - whether you're using comma-separators or not
    - whether you're including <CR><LF>'s or not (or <LF>'s for Linux people) in the record/line
    - whether you're using ASCII or Unicode char's
    - etc.

- You will NEVER HAVE THE ENTIRE dataTable in memory all at once ! ! !