CS3310 – Kaminski - Spr 2015
Asgn 3 Project Specs

World Data App 1.5
CountryData - a hash file on ID
with CodeIndex - a hash table on CODE

DESIGN NOTES

UserApp needs to be able to do: SelectById & SelectByCode
& DeleteById & DeleteByCode & Insert
[So the app needs fast random access on both ID and CODE]
but does NOT need: SelectAllById NOR SelectAllByCode
[So the app does NOT need key-sequential access on either field]

Both id and code uniquely identify countries, so either one could have been used for:
- the PK (PrimaryKey) for structuring the actual DATA file,
- the CK (CandidateKey) for setting up an INDEX for the data file.

[The designer chose id as the PK and country code as the CK]

id values are NOT
- a nice compact set of small numbers
- with a known key-distribution
- with relatively few gaps

[So the app can NOT use the preferred Direct Address file structure with O(1)]

So,
- A hash FILE structure on id is the best choice for the CountryData file
  - An EXTERNAL structure was chosen because of the low activity rate on id and the size of the data records (thus avoiding the costs of the save/load operations)
  - Hash files require fixed-length record locations, and so the file uses fixed-length records and fixed-length fields
- A hash TABLE structure on code is the best choice for CodeIndex
  - An INTERNAL structure was chosen because of the high activity rate on country code and the small size of the index records (thus save/load operations are needed using IndexBackup file)

CountryData.bin and IndexBackup.bin are both now BINARY files rather than a TEXT files
- to save space on numeric fields since we're using fixed-length fields anyway
- and to facilitate handling on the numeric fields

PROJECT A3 OVERVIEW

A3 is a modification of A2. Assume A3 is the same as A2 unless noted in these specs or in class. The only changes needed should be:

- the implementation of CountryData file (all handled inside DataStorage class)
- the implementation of CodeIndex (all handled inside CodeIndex class)
- replace BstNode class with HashNode class
- slight changes to TestDriver program (to be specified in the DemoSpecs)
- slight changes to PrettyPrint program since
  - CountryData & IndexBackup are now binary files (with no <CR><LF>'s)
  - IndexBackup file has different fields than in A2
  - The files have different header record data
- removal of AI and AC options in UserApp's big switch
- remove status message printing

DUMMY STUBS

- DataStorage's delete and CodeIndex' s delete are still dummy stubs – they report the following to uiOutput object (i.e., the Log file)
  ```
  SORRY, deleteById not yet working
  SORRY, deleteByCode not yet working
  ```
- there will be NO INSERT transactions in TransData file for now (so keep that case in UserApp's switch) – so DataStorage's overloaded insert method (#2) should just be a dummy stub for now, which reports the following to uiOutput object:
  ```
  SORRY, insert not yet working
  ```
- DataStorage class still needs an overloaded insert method (#1) to handle Setup's call to it (which sends in the 7 string fields using RawData's getters)
- CodeIndex class needs its insert method

THE 2 BINARY FILES

CountryData.bin and IndexBackup.bin are both BINARY files.
- No field separators (e.g., csv)
- No serialized files
- No <CR><LF> after each record
- shorts are 2 bytes, ints are 4 bytes, longs are 8 bytes, floats are 4 bytes

CountryData file
Header record has 3 shorts: MAX_N_LOC, nHomeRec, nCollRec
Data records:
- code – 3 char's
- id – a short
- name – 18 char's
- continent – 13 char's
- area – an int
- population – a long
- lifeExpectancy – a float

IndexBackup file
Header record has 3 shorts: MAX_N_HOME_LOC, nHomeRec, nCollRec
Data records:
- code – 3 char's
- drp – a short
- hp/link – a short

IMPORTANT:
- RawData provides fields as variable-length STRINGS (when its getters are called)
- DataStorage and CodeIndex methods/setters are responsible for:
  - Converting numeric string fields into short/int/long/float's as needed
  - Padding/truncating alphanumeric fields to designated length as needed

HASHING SPECS

(See separate file describing Hashing Concepts.)

Both CountryData FILE and CodeIndex TABLE use static hashing.

DataStorage & CodeIndex classes MUST each HAVE:

- A named constants for MAX (only named constant used, not 20 or 30)
  - MAX_N_LOC = 30 for DataStorage
  - MAX_N_HOME_LOC = 20 for CodeIndex
- A physically separate method called hashFunction where
  - key value(id or code) AND MAX… is sent in
  - homeAddress is returned

Home Area:

- CountryData FILE: RRNs of 1 to MAX_N_LOC (since RRNs start at 1 not 0).
- CodeIndex TABLE (array)s: subscripts 0 to MAX_N_HOME_LOC (since subscripts start at 0 not 1)

Collision Resolution Algorithms:

- CountryData file: 1) linear (with wrap-around) 2) embedded
- CodeIndex table: 1) chaining 2) separate area

HashFunctions

- CountryData FILE uses the famous "Division-Remainder" with 0-conversion:
  1. divide id by MAX_N_LOC, and use the remainder
     (not the quotient) (HINT: use % operator)
  2. if remainder is 0, change it to MAX_N_LOC
  3. RETURN this homeAddress which is between 1 & 30, inclusive

- CodeIndex TABLE uses this function:
  1. convert 3 char's in code to their 3 ASCII code values
     (giving 3 numbers, all between 65 to 90, inclusive, for A to Z)
  2. multiply those 3 numbers together
     *(giving numbers between 65*65*65=274,625 (for AAA) and 90*90*90=729,000 (for ZZZ))*
  3. use division-remainder algorithm – that is,
     divide step #2 result by MAX_N_HOME_LOC & use remainder
     (giving a number between 0 & MAX_N_HOME_LOC – 1, inclusive)
  4. RETURN homeAddress which is between 0 & 19, inclusive

################################ LOG FILE ################################

No STATUS MESSAGES for file open/close or program start/end

Transaction processing

```
SC CHN   /// visited 2 index nodes
   CHN 094 China            Asia         9,572,900 1,277,558,000 71.4
SI 94    /// visited 4 data file records
   CHN 094 China            Asia         9,572,900 1,277,558,000 71.4
SC WMU   /// visited 3 index nodes
   invalid code
SI 599   /// visited 6 data file records
   invalid id
DI 23
   SORRY, deleteById not yet working
DC ABC
   SORRY, deleteByCode not yet working
IN . . .
   SORRY, insert not yet working
```

PrettyPrint shows

1) CountryData file as it did in A2, except the HeaderRec includes additional fields
2) IndexBackup file prints like this:

```
MAX_N_HOME_LOC: 20, nHome: 17, nColl: 9

LOC     CODE  DRP  HP/LINK
000 >>>  MEX  012  -01
001 >>>  CHN  003  033
002 >>>
. . .                                   (with this part filled in, of course)
027 >>>  OMN  033  -01
028 >>>  DEU  013  032
```

################################ CodeIndex STORAGE ISSUES ################################

Physical storage (in memory):

- 3 short integers: MAX_N_HOME_LOC (constant), nHome (counter), nColl (counter)
- array of hashNodes where a node contains:
  1) code - stored as a string or a 3-char-array
  2) DRP - the RRN of the actual record location in the CountryData FILE.
     This is NOT necessarily the record's homeAddress in the file.
     It's the FINAL ACTUAL STORAGE LOCATION which might be a homeAddress or a collision address.
     DRP is NOT the country's ID, it is its RRN.
  3) Link (for the linked list or chain) – for nodes in:
     a. the HOME area, this is the HP (headPtr) for the collision chain for THIS synonym family, (a -1 if it's an empty chain)
     b. the COLLISION area, this is the link to the next node in this chain

Initialization of physical storage?

- Is this necessary?
  (Yes, for the HOME area, but not for the COLLISION area)
- How are locations "given out"?
  (Based on the calculated homeAddress from the hashFunction and if full, using MAX_N_HOME_LOC + nColl)
- Will it be necessary to test whether a location is empty or not?
  (Yes, for the HOME area, but not for the COLLISION area)
- At the end of Setup, will there be any empty holes in the array?
  (Yes, in the HOME area, but not in the COLLISION area)
- Do nHome and nColl need initializing?
  (Yes, since they're counters)
  (And nColl is used to calculate nextEmpty in the COLLISION area)