# CS 3743 Database
# Program 1: Intro to Hash Files
## DUE DATE: As shown in the Blackboard Assignment



**Converted to Java by Docrob**

This is the first part of a two-part programming assignment. In this part, we are reading and writing to a hashed file. In the next part, we will be implementing probing to handle synonyms.

Files I provide :

**src/java/main** package

**Main.java** - contains the main() entry point for the program, which invokes your functions. It also provides the **hash** function. This class is also called the driver.

**StudentFunctions.java** - this is the file where you will write all of your code

**src/java/hashdb** package

**Vehicle** - the vehicle data model (has all the fields that represent a vehicle in our program)

**HashFile** - contains a reference to the hash file data

**HashHeader** - contains summary information about the hash table

**src/java/misc** package

**ParserException** - an unchecked exception wrapper

**ReturnCodes** - contains all return codes used throughout the program. Preserved from C assignment

**p1Input.txt** - stream input file used by the driver to specify what needs to be invoked

Your hash file will have a header record at Record Block Number (RBN) 0. All other records will either be empty or contain vehicles.

Please see the sample output at the end of this document.

Functions you must code (each of the functions return either RC_OK or an error code):

`public static int hashCreate(String fileName, HashHeader hashHeader)`

This function creates a hash file containing only the HashHeader record.

- If the file already exists, return RC_FILE_EXISTS
- Create the binary file by opening it.
- Write the HashHeader record to the file at RBN 0.
- close the file.
- return RC_OK.

`public static int hashOpen(String fileName, HashFile hashFile)`

This function opens an **existing** hash file which must contain a HashHeader record, and sets the pHashFile->pFile. It returns the HashHeader record by setting pHashFile->hashHeader.

- Determine if the file exists. If not, return RC_FILE_NOT_FOUND.
- Read the HashHeader record from file and return it through the parameter. If the read fails,return RC_HEADER_NOT_FOUND.
- Be sure to set the file inside HashFile to the opened RandomAccessFile
- return RC_OK

`public static int readRec(HashFile hashFile, int rbn, Vehicle vehicle)`

This function reads a record at the specified RBN in the specified file.

- Determine the RBA based on the RBN and the HashHeader's recSize.
- Use the RandomAccessFile seek method to position the file inthat location.
- Read that record and return it through vehicle parameter.
- If the location is not found, return RC_LOC_NOT_FOUND. Otherwise, return RC_OK.
- Note: if the location is found, that does NOT imply that a vehiclewas written to that location. Why?

`public static int writeRec(HashFile hashFile, int rbn, Vehicle vehicle)`

This function writes a record to the specified RBN in the specified file.

- Determine the RBA based on the RBN and the HashHeader's recSize.
- Use the RandomAccessFile seek method to position the file in that location.
- Write that record to the file.
- If the write fails, return RC_LOC_NOT_WRITTEN. Otherwise, return RC_OK.

`public static int vehicleInsert(HashFile hashFile, Vehicle vehicle)`

This function inserts a vehicle into the specified file.

- Determine the RBN using the Main's hash function.
- Use readRec to read the record at that RBN.
- If that location doesn't exist or the record at that location has a vehicleId[0] == "":
  - Write this new vehicle record at that location using writeRec.
- If that record exists and that vehicle's vehicleId matches, return RC_REC_EXISTS. (Do not update it.)
- Otherwise, return RC_SYNONYM. Note that in program #2, we will actually insert synonyms.

`public static int vehicleRead(HashFile hashFile, int rbn, Vehicle vehicle)`

This function reads the specified vehicle by its vehicleId.

- Since the vehicles vehicleId was provided, determine the RBN using the Main's hash function.
- Use readRec to read the record at that RBN.
- If the vehicle at that location matches the specified vehicle's vehicleId, return the vehicle via the vehicle parameter and return RC_OK.
- Otherwise, return RC_REC_NOT_FOUND

Note:

1. Your code must be written in the **StudentFunctions.java** file
2. Do NOT modify ANY of the other files.
3. TO SUBMIT: zip your src directory and the pom.xml in a single archive file

**THIS IS A MAVEN PROJECT**

3 different variations of the starter project have been provided for you. Pick one:

1. A basic maven project set up for JDK 11 that is IDE independent
2. An IntelliJ maven project set up for JDK 11
3. An Eclipse maven project set up for JDK 8 (the JDK that is available on UTSA's VDI)

JDK 11 is recommended, however JDK 8 will work fine. To switch the JDK version in the maven project, change the 2 integers in the properties section of the pom.xml file. E.g., to change from JDK 11 to JDK 8, change the two "11"s to "8"s in the pom.xml file. You should then type "mvn clean" to remove all previous target files.

I highly recommend using maven from the command line to build and run this program. I personally use IntelliJ to work with the code, but use maven from IntelliJ's integrated terminal to build and execute. There is a ton of info about maven on the internet. Start here: hdps://maven.apache.org/what-is-maven.html

**To compile your code from the directory above cs3743 (package directory):**

```
mvn compile
```

**To build (includes compiling) and execute your program:**

```
mvn package exec:java
```

**SAMPLE OUTPUT**

Please see the sample output starting on the next page

**GRADING RUBRIC**

2 points for each of the ten sections of the input file

Sample Output (partial):

```
* CS3743 p1Input.txt
* Nuke the Hash file if it exists using driver function
 >> NUKE VEHICLE vehicle.dat
* Opening of a non-existent file should cause an error
 >> OPEN VEHICLE vehicle.data
    **** ERROR: file not found or invalid header record
*
* 1. Create the hash file
*
>> CREATE VEHICLE vehicle.dat 19 5
    Record size is 72
>> PRINTALL VEHICLE vehicle.dat
    MaxHash=19, RecSize=72, MaxProbes=5
*
* 2. Creating it again should cause an existence error
*
 >> CREATE VEHICLE vehicle.dat 17 5
     Record size is 72
     **** ERROR: file already
    exists
* Open it
 >> OPEN VEHICLE vehicle.dat
*
* 3. Insert some vehicles
*
 >> INSERT VEHICLE CB001,CHEVY,BelAir,1957
             Hash RBN is 13
 >> PRINTALL VEHICLE vehicle.dat
     MaxHash=19, RecSize=72, MaxProbes=5
     13  CB001 1957 CHEVY             BelAir Hash=13

 >> INSERT VEHICLE DD001,DODGE,Dart,1970
             Hash RBN is 16
 >> PRINTALL VEHICLE vehicle.dat
     MaxHash=19, RecSize=72, MaxProbes=5
     13  CB001 1957 CHEVY             BelAir Hash=13
     16  DD001 1970 DODGE               Dart Hash=16
*
* 4. Read an existing vehicle
*
 >> READ VEHICLE CB001
             Hash RBN is 13
     13  CB001 1957 CHEVY             BelAir Hash=13
*
* 5. Read should not find these two vehicles
*
 >> READ VEHICLE TP001
             Hash RBN is 6
     **** ERROR: record not found
 >> READ VEHICLE FE001
             Hash RBN is 19
     **** ERROR: record not found

*
* 6. Insert two more vehicles
*
 >> INSERT VEHICLE TP001,TOYOTA,Prius,2007
             Hash RBN is 6
 >> INSERT VEHICLE DD002,DODGE,Dart,1974
             Hash RBN is 17
 >> PRINTALL VEHICLE vehicle.dat
     MaxHash=19, RecSize=72, MaxProbes=5
      6  TP001 2007 TOYOTA              Prius Hash=6
     13  CB001 1957 CHEVY             BelAir Hash=13
     16  DD001 1970 DODGE               Dart Hash=16
```

```
     17  DD002 1974 DODGE               Dart Hash=17
*
* 7. Insert an existing vehicle - should cause an error
*
 >> INSERT VEHICLE CB001,CHEVY,BelAir,1966
             Hash RBN is 13
     **** ERROR: record already exists
 >> READ VEHICLE CB001
             Hash RBN is 13
     13  CB001 1957 CHEVY             BelAir Hash=13
```