PS 2: Code-reading and design questions

The Table and Column classes

1) **open()** method allows us to open already-exist table and we can access the table by SQL command. If the table does not exist, it would print a string in a format of **<table_name>: no such table** and return an operation status of **NOTFOUND**. With that return value, we should throw an exception.
2) **primaryKeyColumn()** determines whether a table has a primary key.
3) **getColumn(int)** allows us to obtain columns and **getColumn(0)** allows us to obtain the leftmost column.
4) **getType()** method allows to determine the type of the column. The possible return values are int:
    a) 0 for INTEGER
    b) 1 for REAL
    c) 2 for CHAR
    d) 3 for VARCHAR
5) **getLength()** method allows us to determine the length of the values. For VARCHAR, it returns the maximum number of bytes

The SQLStatement class and its subclasses

6) **getTable(int)** method allows to determine which table SQL commands should operate on.
7) Those methods include:
    a) **getColumn(int):** return the column with specified index
    b) **getColumnVal(int):** return column value with specified index
    c) **getWhereColumn(int):** return column with specified index from WHERE clause
    d) **getWhere():** return the WHERE clause as a ConditionalExpression object
    e) **numWhereColumns()**: return the number of columns in the WHERE clause
    f) **numColumnVals():** return the number of column values
    g) **numTables()**: return the number of tables
    h) **numColumns():** return the number of columns

Marshalling

8) **-2** for primary key offset
9) **-1** for NULL value offset
10) Here is the description:
    a) For the first row:
        - key: 1
        - value: -2, 6, 11, hello
    b) For the second row:
        - key: 2
        - value: -2, -1, 6
11) Here is the description:
    a) For the first row:

    - key: 1234
    - value: 10, -2, 14, 19, 27, 1, hello, 12.5
  b) For the second row:
    - key: 4567
    - value: 10, -2, 14, -1, 23, 2, wonderful

12) The methods we would be using are:
  a. **writeShort(int):** allow us to write offsets as 2-bytes int
  b. **writerChars(String):** allow us to write CHAR and VARCHAR type value
  c. **writeInt(int):** allow us to write INTEGER type value
  d. **writeDouble(double):** allow us to write REAL type value
  e. **size():** allow us to know the number of bytes of the data stream

Unmarshalling

13) Firstly, we should create the RowInput object.Then, we use RowInput methods to access the offsets of the particular column value. With the offsets, we can access the value

14) The methods include:
  a. readBooleanAtOffset(int)
  b. readNextBoolean()
  c. readByteAtOffset(int)
  d. readNextByte()
  e. readShortAtOffset(int)
  f. readNextShort()
  g. readIntAtOffset(int)
  h. readNextInt()
  i. readDoubleAtOffset(int)
  j. readNextDouble()
  k. readBytesAtOffset(int, int)
  l. readNextBytes(int)

15) To determine the length of the VARCHAR attribute, first we need to read the starting offset of the value using **readNextShort()**. Then, we read to the ending offset using **readNextShort()**. By, subtracting the two values, we get the length of VARCHAR value

Miscellaneous

16) The execute() method uses:
  a. **new insertRow(Table, Object[])** constructor to construct an InsertRow object for a row containing the to-be-inserted values
  b. **marshall()** method to marshall the collection of values in the InsertRow object to a key/ value pair

17)
  a. First, I need to get the number of bytes and bytes array containing bytes in the RowOutput objects of both key and value with getBufferBytes() and getBufferLength() methods.

   b. Then, we create DatabaseEntry objects for both key and value using **DatabaseEntry constructor**.
   c. Next, we connect to database using database handle with **getDB()** method in Table object
   d. Lastly, we use putNoOverwrite() to perform the insertion

18) Other methods needed are **getColumnVal(int)**