Writeup for Lab 1

姓名：陈志聪　　学号：521120910256

Decision Made for Lab 1

- Exercise 1

  class `TupleDesc`:

  - Added attributes: `tupledesc`, `fields`, `recordid`. `fields` is an `ArrayList` storing `TDItem`, a kind of structure defined in `TupleDesc`. But we can exploit hashmap instead of ArrayList for faster query.

  class `Tuple`:

  - Added attributes: `tupledesc`, `fields`, `recordid`.

- Exercise 2

  class `Catalog`:

  - Added attributes: `file`, `name`, `pkeyField`.
  - Modified methods of importance: most of functions to achieve in this exercise require to handle exception when the input is invalid. Hence, I concentrated on handle invalid index or table name.

- Exercise 3

  class `BufferPool`:

  - Added attributes: `maxNumPages`, `pages`. Specifically, `pages` is a `ConcurrentHashMap`, for faster query and transaction safe.

- Exericse 4

  class `HeapPageId`:

  - Added attributes: `tableId`, `pageNo`

  class `Recordid`:

  - Added attributes: `pageid`, `tuplenumber`

  class `HeapPage`:

  - Added attributes: `pid`, `td`, `header`, `tuples`, `numSlots`
  - Modified methods of importance: `isSlotUesd`. To identify whether i th slot is used, I first identified the corresponding bit in the header(**big-endian** is used in java virtual machine) and then judged with the bit.

- Exercise 5

  class `HeapFile`:

  - Added attributes: `file`, `tupleDesc`
  - Modified methods of importance:
    - `ReadPage`: get page according to input `PageId`. I exploited `RandomAccessFile` to avoid extreme case and focused on read exception when `PageId` is invalid.

        - `iterator`: This function returns iterator of all tuples of each pages in the file. To achieve it, I defined an inner class `HeapFileIterator` which implements `DbFileIterator` API. Since all pages are required to traversed, attribute `curPageNo` is used in this class to record the number of page being traversed. I overrode methods `open()`, `hasNext()`, `rewind()`, `close()`, `next()` and added `getPageTuples()` to get tuple iterator of each page.

- Exercise 6

  class `SeqScan`:

    - Added attributes: `transactionId`, `tableId`, `tableAlias`, `fileIter`, `tupleDesc`.
    - Modified/Added methods of importance:
        - `attachAlias`: a private method that returns a `tupleDesc` in which alias is added to all string names as prefix.

Changes Made to the API

I've not made any changes to any API.

Missing or Incomplete Elements

There may be two elements I've ignored when achieving for the database.

- Time complexity. In some function, to find out the target element, I traversed the whole list or array with time complexity of $O(N)$. Instead, elements can be stored in a hashmap for faster query.
- Exception handling: I did consider exception like `IndexOutOfBoundException` and `NoSuchElementException` in some places. However, I may ignore to judge whether the input index or string is valid or not in `HeapFile.java`.

Timeline and Difficulty

I spent about 2 days on lab 1 and allocated 1day to exercise 1-4 and 1day to exercise 5-6, respectively. During the lab, I found the most difficult part is to figure out structure of the whole database and interface of each class. Besides, I struggled to achieve function `HeapFileIterator` in `HeapFile.java`, which is responsible for return an iterator over all tuples in the file. After coding of all exercise, I failed the `systemtest` for many times, particularly the `testCache()`. There were always one more time when counting the number of readPage operations. After trials and trails, I modified method `hasNext()` in the inner class `HeapFileIterator` and finally passed the test.