Lab1 answer

1. Understanding of Lab1

Tuple:

Main(): store Fields in the class. The schema of it will implement in TupleDesc, we will not directly get from Tuple.

TupleDesc:

Main():represent the schema of target tuple. So if we want to access the attributes of tuple, we will use the method here to get data. For instance, ToString() will help us know what is in the tuple.

ToString(): We are required to present the schema in special format. Here I implement by for loop to do it.

Merge(): using for loop to buildup new Types[] and String[] to make new TDitems of Tuple.

Catalog:

Main(): help HeapFile to track the Tuple.

BufferPool:

Main() Using ConcurrentHashMap to connect PageId and Page makes access Page makes Heapfile can read the page in the bufferpool.

HeapPage:

Main(): represent the page in HeapFile.

isSlotUsed(): using bit operator to track slot whether use.

First, track the i/8 to be index, because 1 int can store 8 byte.
Second, operator>> i%8 to get the target is used or not(for example, No.0 page will be the first header, and the first byte of the int->0000000 means not using.)and & with byte operator to avoid minus value error.

RecordId:

Main(): we can use this to find the specific tuple and Page in RecordId.

HeapFile:

Main(): In the course, we know it will iterating the tuple in the bufferpool. If request from Seqcan is not in the pool, it will require Bufferpool to read the target tuple by tid which given by Seqcan.
readPage(): using randomaccess in Java. The position of data will be at Pagesize() X Number of page. Then we can finish this by using this.

Seqcan:

Main():As we learned in class, the Sequential scan access method that reads table. Therefore, I add the Tableiterator here.
GetTupleDesc():
3. In my opinion, I think the unit test for bufferpool will be useful, because it is hard for us to distinguish crossover files whether impact by error from bufferpool. For instance, when I tried to pass Scantest, I failed at testSmall. Finally, I found I did not complete the part of return the correct Database file in bufferpool. Therefore, I recommend to add the test for getPage() method.
4. I did not modify API.
5. I am not sure fail of TestUtil causes by my missing or its original code.

```
honkuro@MSI:/mnt/d/University of  Washington -seattle/UW course/CSE444/simple-db-honkuro$ ant runtest -Dtest=TestUtil
Buildfile: /mnt/d/University of  Washington -seattle/UW course/CSE444/simple-db-honkuro/build.xml

compile:
    [depend] The class simpledb.HeapFileIterator in file /mnt/d/University of  Washington -seattle/UW course/CSE444/simple-db-honkuro/bin/src/simpledb/HeapFileIterator.class is out of date due to simpledb.Buffe
Pool but has not been deleted because its source file could not be determined
    [depend] Deleted 7 out of date files in 0 seconds
     [javac] Compiling 6 source files to /mnt/d/University of  Washington -seattle/UW course/CSE444/simple-db-honkuro/bin/src

testcompile:

runtest:
     [junit] Running simpledb.TestUtil
     [junit] Testsuite: simpledb.TestUtil
     [junit] Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.014 sec
     [junit] Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.014 sec
     [junit]
     [junit] Testcase: initializationError took 0.002 sec
     [junit]     Caused an ERROR
     [junit] No runnable methods
     [junit] java.lang.Exception: No runnable methods
     [junit]     at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
     [junit]
```