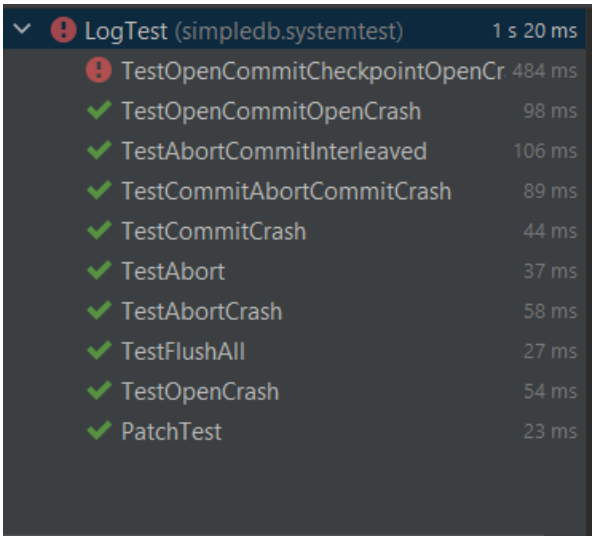
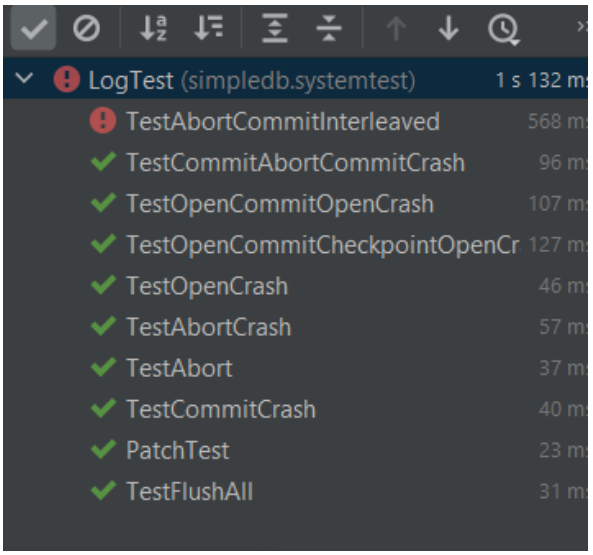
I added a log for this DBMS, this log can record the activities of each transaction. Thus, if a transaction was aborted or the whole database crashed, we can recover it by using this log. My rollback and recovery are straightforward. When rollback a transaction, just find where the transaction starts and reset the pages that have not been reset. When recovering the database, rollback all uncommitted transactions.

Discuss and justify any changes you made outside of LogFile.java:
I made some changes in BufferPool and LockManager classes.
For the LockManager Class, I made its inner class Lock become public, and added a function to return the “transactionLockList”. By doing this, BufferPool now can access the Locks that each Transactions hold.
For BufferPool Class, in addition to making evictPage to be able to evict dirty pages, I also made the transactionComplete function to get dirty pages from LockManager. Before the modification my transactionComplete function was reading dirty from BufferPool, but I think there may be a synchronization problem doing this. So I changed this part and that's why I changed the LockManager class as well.

Describe a unit test that could be added to improve the set of unit tests that we provided for this lab.

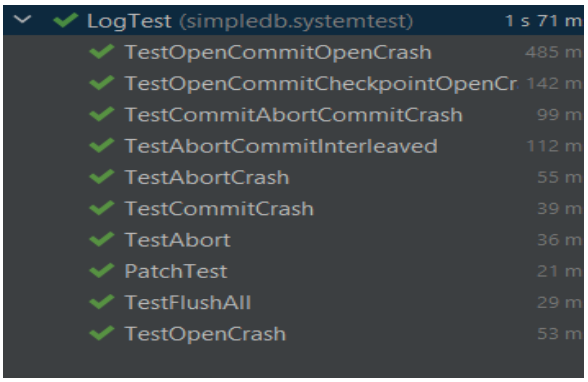
When I tested my code I found that as long as TestAbortCommitInterleaved and TestOpenCommitCheckpointOpenCrash were the first test the code ran, they must fail.

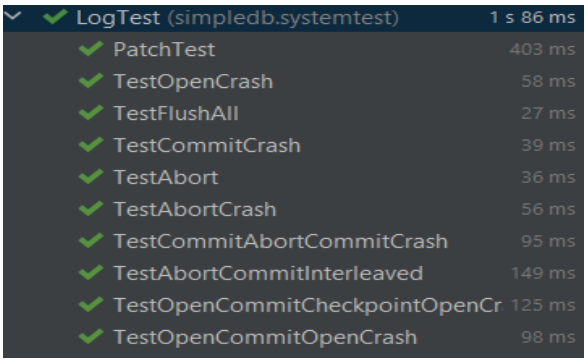


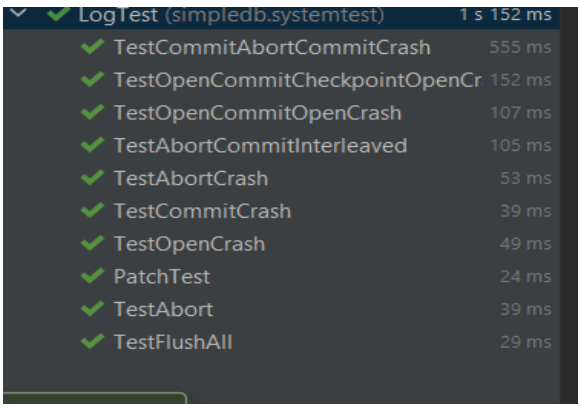


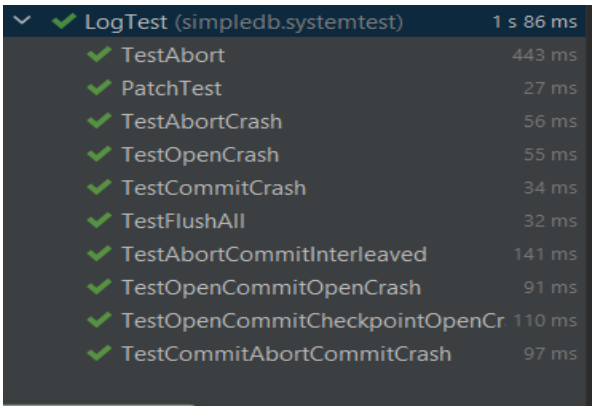
(Next page)

But if the code runs other tests first, I can pass all tests.









I tried to debug and found that “TestOpenCommitCheckpointOpenCrash” seems to reset the whole logFile after it is called `Database.getLogFile().logCheckpoint();`.
To let it print the content of logFile before and after logCheckpoint():

```
@Test public void TestOpenCommitCheckpointOpenCrash()
        throws IOException, DbException, TransactionAbortedException {
    setup();
    doInsert(hf1, t1: 1, t2: 2);
    // *** Test:
    // T1 inserts but does not commit
    // T2 inserts and commits
    // checkpoint
    // T3 inserts but does not commit
    // crash
    // only T2 data should be there
    Transaction t1 = new Transaction();
    t1.start();
    insertRow(hf1, t1, v1: 12, v2: 0);
    Database.getBufferPool().flushAllPages(); // XXX defeat NO-STEAL-based abort
    insertRow(hf1, t1, v1: 13, v2: 0);

    // T2 commits
    doInsert(hf2, t1: 26, t2: 27);

    System.out.println("LogFile before logCheckpoint() was called -------------------------");
    Database.getLogFile().print();
    Database.getLogFile().logCheckpoint();
    System.out.println("LogFile after logCheckpoint() was called -------------------------");
    Database.getLogFile().print();
```

Here is the output:

```
LogFile before logCheckpoint() was called -------------------------
file length: 50052
lastCKPT: -1
no CKPT was found

Log after lastCKPT:
BEGIN, TID: 0, log offset: 8
UPDATE, TID: 0, page ID: simpledb.HeapPageId@80000000, log offset: 28
UPDATE, TID: 0, page ID: simpledb.HeapPageId@80000000, log offset: 8352
COMMIT, TID: 0, log offset: 16676
BEGIN, TID: 1, log offset: 16696
UPDATE, TID: 1, page ID: simpledb.HeapPageId@80000000, log offset: 16716
BEGIN, TID: 2, log offset: 25040
UPDATE, TID: 1, page ID: simpledb.HeapPageId@80000000, log offset: 25060
UPDATE, TID: 2, page ID: simpledb.HeapPageId@80000000, log offset: 33384
UPDATE, TID: 2, page ID: simpledb.HeapPageId@80000000, log offset: 41708
COMMIT, TID: 2, log offset: 50032
LogFile after logCheckpoint() was called -------------------------
file length: 0

java.io.EOFException Create breakpoint
    at java.base/java.io.RandomAccessFile.readInt(RandomAccessFile.java:837)
    at java.base/java.io.RandomAccessFile.readLong(RandomAccessFile.java:870)
    at simpledb.LogFile.print(LogFile.java:603)
    at simpledb.systemtest.LogTest.TestOpenCommitCheckpointOpenCrash(LogTest.java:408)
```

Before it was called logCheckpoint(), the size of logFile was 50052, but became 0 after it was called. Because logFile size became 0, the raf.readLong() in my print method threw an exception.

I don’t know why this happened, and if you can make something to improve the tests.