ReadMe File

Team members	Unity Id
Venkata Sai Prashanth Rallapalli	vrallap
Abinav Pothuganti	apothug
Sri Harsha Kunapareddy	skunapa
Shiv Shankar Barai	sbarai

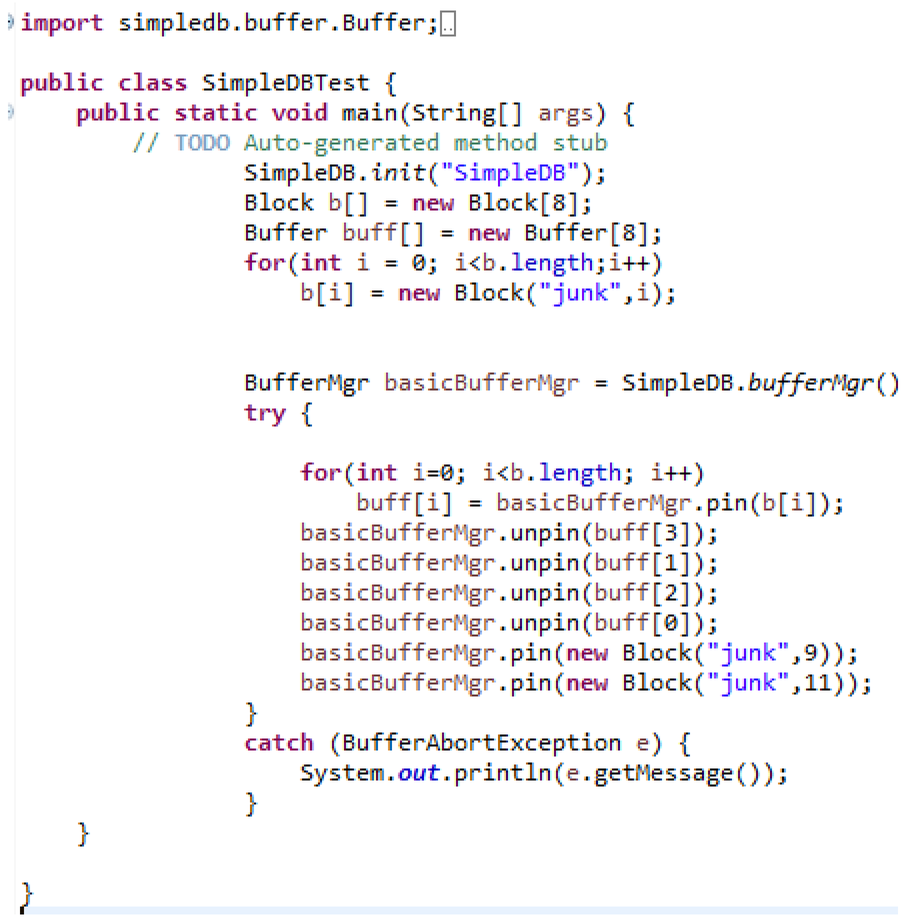
List of files changed:

1) For TASK1:
   a) *BasicBufferMgr.java* in simpledb.buffer package
2) For TASK2:
   a) *LogIterator.java* in simpledb.log package
   b) *LogMgr.java* in simpledb.log package
   c) *CheckPointRecord.java* in simpledb.tx.recovery
   d) *CommitRecord.java* in simpledb.tx.recovery package
   e) *logRecord.java* in simpledb.tx.recovery package
   f) *LogRecordIterator.java* in simpledb.tx.recovery package
   g) *RecoveryMgr.java* in simpledb.tx.recovery package
   h) *RollbackRecord.java* in simpledb.tx.recovery package
   i) *SetIntRecord.java* in simpledb.tx.recovery package
   j) *SetStringRecord.java* in simpledb.tx.recovery package
   k) *StartRecord.java* in simpledb.tx.recovery package

Task 1: Buffer manager

Testing program 1: simpleDBTest.java

Run simpleDBTest.java



```
import simpledb.buffer.Buffer;

public class SimpleDBTest {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
                SimpleDB.init("SimpleDB");
                Block b[] = new Block[8];
                Buffer buff[] = new Buffer[8];
                for(int i = 0; i<b.length;i++)
                    b[i] = new Block("junk",i);


                BufferMgr basicBufferMgr = SimpleDB.bufferMgr()
                try {

                    for(int i=0; i<b.length; i++)
                        buff[i] = basicBufferMgr.pin(b[i]);
                    basicBufferMgr.unpin(buff[3]);
                    basicBufferMgr.unpin(buff[1]);
                    basicBufferMgr.unpin(buff[2]);
                    basicBufferMgr.unpin(buff[0]);
                    basicBufferMgr.pin(new Block("junk",9));
                    basicBufferMgr.pin(new Block("junk",11));
                }
                catch (BufferAbortException e) {
                    System.out.println(e.getMessage());
                }
    }

}
```

Functionality:

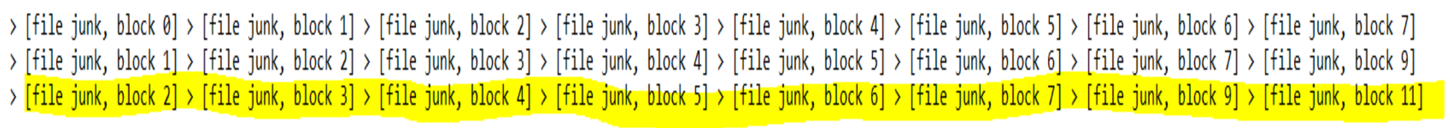
1. 8 blocks and buffers are created.

2. Each block is pinned to a buffer.

3. Some of the buffers are unpinned and tried to pin new blocks to the buffers.

Output:

The replacement policy implemented here is FIFO.



```
› [file junk, block 0] › [file junk, block 1] › [file junk, block 2] › [file junk, block 3] › [file junk, block 4] › [file junk, block 5] › [file junk, block 6] › [file junk, block 7]
› [file junk, block 1] › [file junk, block 2] › [file junk, block 3] › [file junk, block 4] › [file junk, block 5] › [file junk, block 6] › [file junk, block 7] › [file junk, block 9]
› [file junk, block 2] › [file junk, block 3] › [file junk, block 4] › [file junk, block 5] › [file junk, block 6] › [file junk, block 7] › [file junk, block 9] › [file junk, block 11]
```

Task 2:

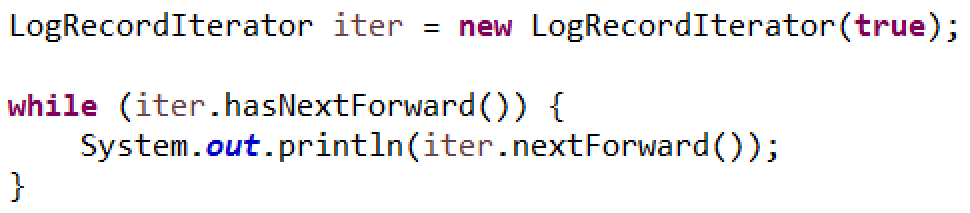
Methods Added:

```
/**
        * Moves to the next log record in forward order.
        * @return the next earliest log record in forward direction
        */

public LogRecord nextForward() {
             LogRecord lr = records.get(0);
             records.remove(0);
             return lr;
      }

/**
    * Moves to the next log block in forward order,
    * and positions at the first record in that block.
    */
   private void moveToNextForwardBlock() {
      blk = new Block(blk.fileName(), blk.number()+1);
      pg.read(blk);
      currentrec = INT_SIZE;
   }
```

Made changes to the following methods:

```
/**
    * Writes a setint record to the log, and returns its lsn.
    * Updates to temporary files are not logged; instead, a
    * "dummy" negative lsn is returned.
    * @param buff the buffer containing the page
    * @param offset the offset of the value in the page
    * @param newValue the value to be written
    */
   public int setInt(Buffer buff, int offset, int newValue) {
      int oldValue = buff.getInt(offset);
      Block blk = buff.block();
      if (isTempBlock(blk))
         return -1;
      else
         return new SetIntRecord(txnum, blk, offset, oldValue,
newValue).writeToLog();
   }

   /**
    * Writes a setstring record to the log, and returns its lsn.
    * Updates to temporary files are not logged; instead, a
    * "dummy" negative lsn is returned.
    * @param buff the buffer containing the page
    * @param offset the offset of the value in the page
    * @param newValue the value to be written
    */
   public int setString(Buffer buff, int offset, String newValue) {
      String oldValue = buff.getString(offset);
      Block blk = buff.block();
      if (isTempBlock(blk))
         return -1;
      else
         return new SetStringRecord(txnum, blk, offset, oldValue,
newValue).writeToLog();
   }
```

Part1: LogRecordIterator



```java
LogRecordIterator iter = new LogRecordIterator(true);

while (iter.hasNextForward()) {
    System.out.println(iter.nextForward());
}
```

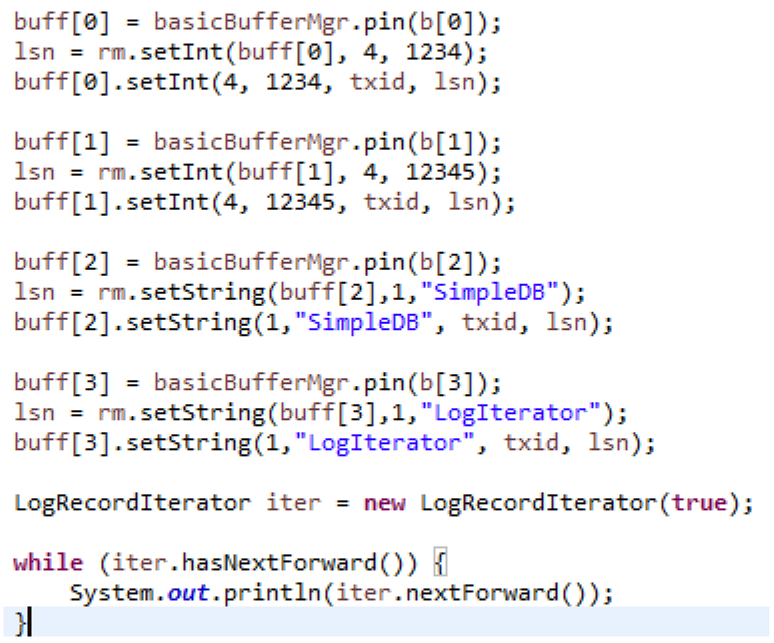
Functionality:

1. Created a block and pinned it to a buffer.
2. Created a recovery manager for a transaction.
3. Used setInt and setString to set logs for the transaction.
4. Used iter.next() to print the logs are read in forward manner and prints old and new value.
5. Used multiple blocks and repeated the above steps.

Running the **LogRecordIteratorTest.java** file in the default package executes this test and the output can be seen in the console.

Output:

Sample output for



```java
buff[0] = basicBufferMgr.pin(b[0]);
lsn = rm.setInt(buff[0], 4, 1234);
buff[0].setInt(4, 1234, txid, lsn);

buff[1] = basicBufferMgr.pin(b[1]);
lsn = rm.setInt(buff[1], 4, 12345);
buff[1].setInt(4, 12345, txid, lsn);

buff[2] = basicBufferMgr.pin(b[2]);
lsn = rm.setString(buff[2],1,"SimpleDB");
buff[2].setString(1,"SimpleDB", txid, lsn);

buff[3] = basicBufferMgr.pin(b[3]);
lsn = rm.setString(buff[3],1,"LogIterator");
buff[3].setString(1,"LogIterator", txid, lsn);

LogRecordIterator iter = new LogRecordIterator(true);

while (iter.hasNextForward()) {
    System.out.println(iter.nextForward());
}
```

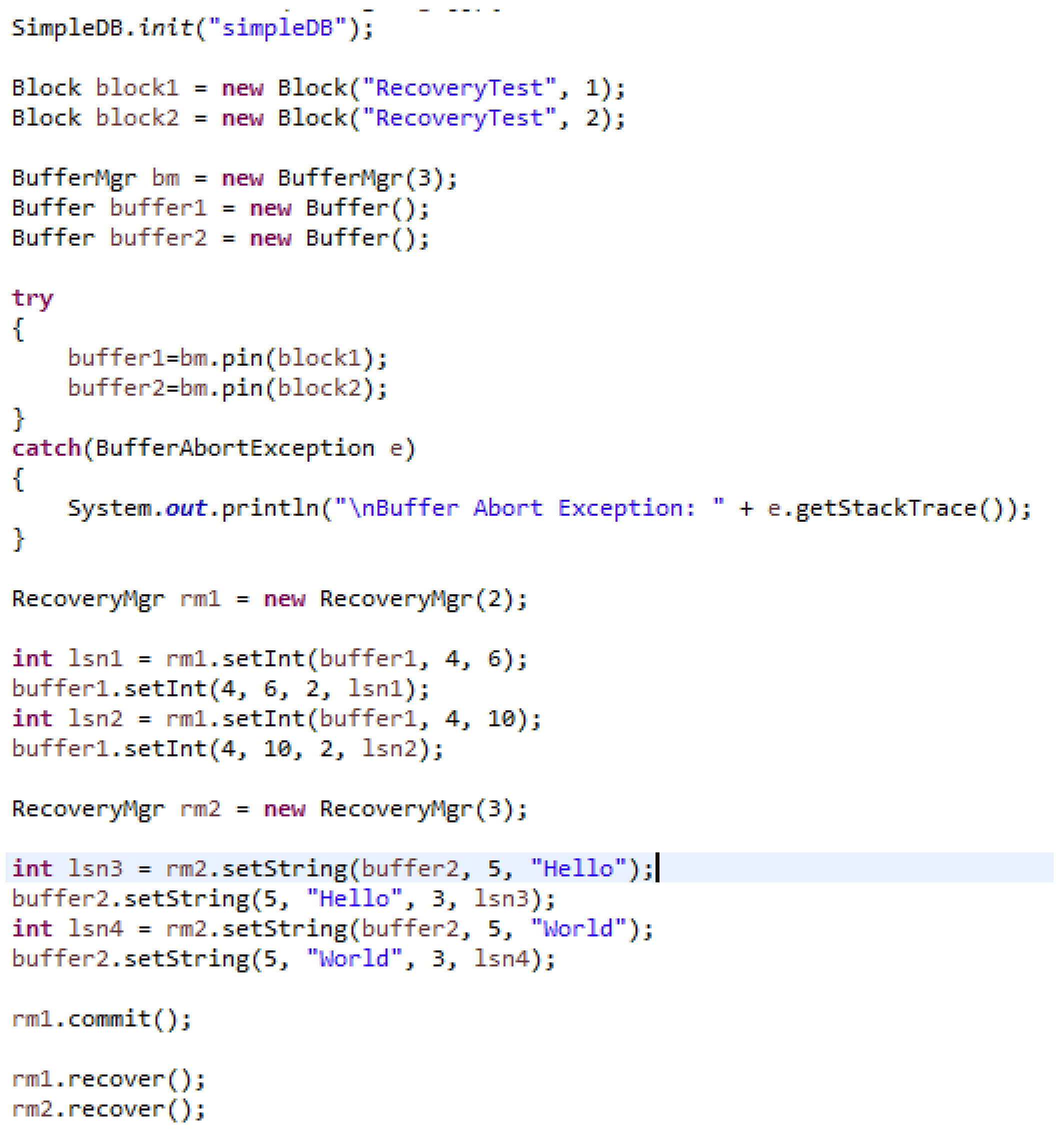
Is shown below

```
<START 2>
<SETINT 2 [file LogIteratorRecoveryTest, block 0] 4 0 1234>
<SETINT 2 [file LogIteratorRecoveryTest, block 1] 4 0 12345>
<SETSTRING 2 [file LogIteratorRecoveryTest, block 2] 1  SimpleDB>
<SETSTRING 2 [file LogIteratorRecoveryTest, block 3] 1  LogIterator>
```

Part 2: Recovery

Testing program 2: RecoveryTesting.java

Run RecoveryTest class as java application.



```java
SimpleDB.init("simpleDB");

Block block1 = new Block("RecoveryTest", 1);
Block block2 = new Block("RecoveryTest", 2);

BufferMgr bm = new BufferMgr(3);
Buffer buffer1 = new Buffer();
Buffer buffer2 = new Buffer();

try
{
    buffer1=bm.pin(block1);
    buffer2=bm.pin(block2);
}
catch(BufferAbortException e)
{
    System.out.println("\nBuffer Abort Exception: " + e.getStackTrace());
}

RecoveryMgr rm1 = new RecoveryMgr(2);

int lsn1 = rm1.setInt(buffer1, 4, 6);
buffer1.setInt(4, 6, 2, lsn1);
int lsn2 = rm1.setInt(buffer1, 4, 10);
buffer1.setInt(4, 10, 2, lsn2);

RecoveryMgr rm2 = new RecoveryMgr(3);

int lsn3 = rm2.setString(buffer2, 5, "Hello");
buffer2.setString(5, "Hello", 3, lsn3);
int lsn4 = rm2.setString(buffer2, 5, "World");
buffer2.setString(5, "World", 3, lsn4);

rm1.commit();

rm1.recover();
rm2.recover();
```

Functionality:

1. Multiple blocks and buffers are created.

2. Try to pin blocks to buffers.

a) If buffer is full Buffer Abort Exception is thrown by buffer manager

b) Else blocks get pinned to buffers

3. Recovery manager objects are created.

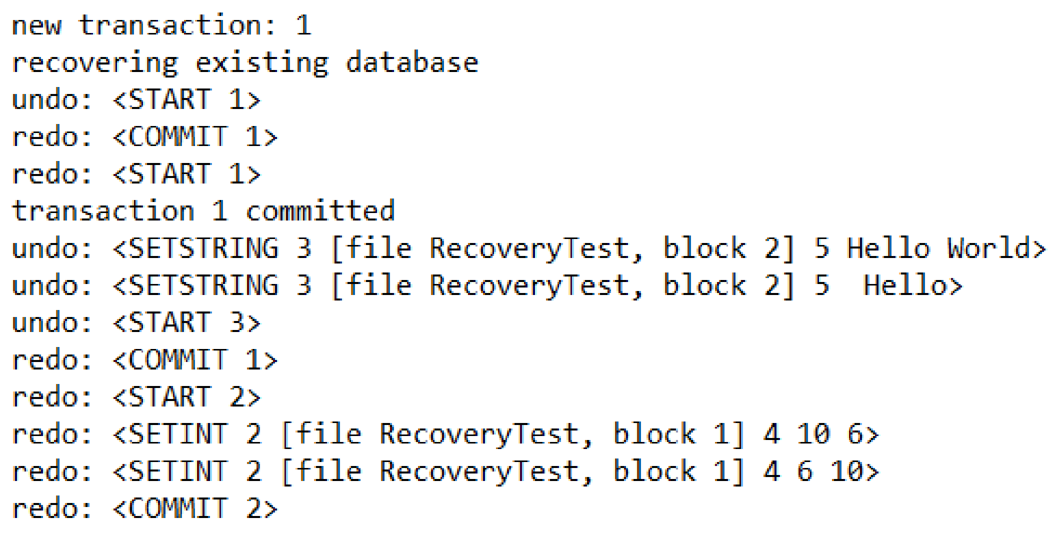
4. More than one Integer and String values are Inserted in each buffer at given offsets using setInt() and setString() methods respectively.

5. Transaction is commited.

Output:

1. In undo phase Log record should be updated with old value

2. After Undo, Redo phase should be triggered and value should be updated with new value.



```
new transaction: 1
recovering existing database
undo: <START 1>
redo: <COMMIT 1>
redo: <START 1>
transaction 1 committed
undo: <SETSTRING 3 [file RecoveryTest, block 2] 5 Hello World>
undo: <SETSTRING 3 [file RecoveryTest, block 2] 5  Hello>
undo: <START 3>
redo: <COMMIT 1>
redo: <START 2>
redo: <SETINT 2 [file RecoveryTest, block 1] 4 10 6>
redo: <SETINT 2 [file RecoveryTest, block 1] 4 6 10>
redo: <COMMIT 2>
```