Exercise1

Implementation of Tupe.java

public String toString() Overrides the toString method to represent the contents of the tuple as a string. It concatenates the string representation of each field, separating them with tabs (\t).

public void resetTupleDesc(TupleDesc td) Resets the TupleDesc of the tuple (affects only the TupleDesc).Updates the td member variable with the provided TupleDesc. Initializes the tupleFields ArrayList with null elements based on the number of fields in the new TupleDesc.

Implementation of TupeDesc.java

```
testcompile:

runtest:
    [junit] Running simpledb.TupleDescTest
    [junit] Testsuite: simpledb.TupleDescTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.057 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.057 sec
    [junit]
    [junit] Testcase: getType took 0.002 sec
    [junit] Testcase: numFields took 0.001 sec
    [junit] Testcase: combine took 0.01 sec
    [junit] Testcase: testEquals took 0.002 sec
    [junit] Testcase: nameToId took 0.016 sec

BUILD SUCCESSFUL
Total time: 1 second
```

```
runtest:
    [junit] Running simpledb.TupleTest
    [junit] Testsuite: simpledb.TupleTest
    [junit] Tests run: 3, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.038 sec
    [junit] Tests run: 3, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.038 sec
    [junit]
    [junit] Testcase: getTupleDesc took 0.017 sec
    [junit] Testcase: modifyRecordId took 0.007 sec
    [junit]     Caused an ERROR
    [junit] modifyRecordId() test failed due to RecordId.equals() not being implemented.  This is not required for Lab 1, bu
t should pass when you do implement the RecordId class.
    [junit] java.lang.UnsupportedOperationException: modifyRecordId() test failed due to RecordId.equals() not being impleme
nted.  This is not required for Lab 1, but should pass when you do implement the RecordId class.
    [junit]     at simpledb.TupleTest.modifyRecordId(TupleTest.java:56)
C:\Users\annaa\Desktop\Lab Database\project_24\build.xml:63: Test simpledb.TupleTest failed
```

Exercise2

Implementation of Catalog.java

Two ConcurrentHashMap objects are used to store information about tables. **catalog_map** is indexed by the table ID, and **nameID_map** is indexed by the table name.

public void loadSchema reads the schema from a file and creates the appropriate tables in the database. The code primarily focuses on maintaining a catalog of tables, providing methods to add tables, retrieve information about tables, and load table schemas from a file. It's designed to be thread-safe with the use of ConcurrentHashMap for storing table information.

```
runtest:
    [junit] Running simpledb.CatalogTest
    [junit] Testsuite: simpledb.CatalogTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.033 sec
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.033 sec
s took 0.002 sec
    [junit] Testcase: getDatabaseFile took 0.001 sec
    [junit] Testcase: handleDuplicateNames took 0 sec

BUILD SUCCESSFUL
Total time: 2 seconds
```

Exercise 3

Implementation of BufferPool.java

Variable Instance **bufferPool** is A HashMap that represents the buffer pool, mapping PageId to Page objects.

getPage method retrieves a page from the buffer pool or fetches it from disk if not present.
It checks if the page is already in the buffer pool. If yes, it returns the page. If the page is not in the buffer pool, it fetches it from the database file. It checks if there is sufficient space in the buffer pool. If full, it throws a DbException. If there is space, it adds the fetched page to the buffer pool and returns it.

Exercise 4

```
testcompile:

runtest:
    [junit] Running simpledb.HeapPageId
Test
    [junit] Testsuite: simpledb.HeapPag
eIdTest
    [junit] Tests run: 4, Failures: 0,
Errors: 0, Skipped: 0, Time elapsed: 0.024 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.024 sec
    [junit]
    [junit] Testcase: equals took 0.012 sec
    [junit] Testcase: pageno took 0.001 sec
    [junit] Testcase: getTableId took 0 sec
    [junit] Testcase: testHashCode took 0 sec

BUILD SUCCESSFUL
```

```
compile:

testcompile:

runtest:
    [junit] Running simpledb.RecordIdTest
    [junit] Testsuite: simpledb.RecordIdTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.023 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.023 sec
    [junit]
    [junit] Testcase: equals took 0.01 sec
    [junit] Testcase: tupleno took 0.002 sec
    [junit] Testcase: hCode took 0.001 sec
    [junit] Testcase: getPageId took 0 sec

BUILD SUCCESSFUL
Total time: 1 second
```

```
compile:

testcompile:

runtest:
    [junit] Running simpledb.HeapPageReadTest
    [junit] Testsuite: simpledb.HeapPageReadTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 sec
    [junit]
    [junit] Testcase: getSlot took 0.021 sec
    [junit] Testcase: getId took 0.003 sec
    [junit] Testcase: getNumEmptySlots took 0.001 sec
    [junit] Testcase: testIterator took 0.002 sec

BUILD SUCCESSFUL
Total time: 1 second
```

Implementation of HeapPage.java

We implemented the iterator() method for the HeapPage class. Custom Iteratorclass iterates over the tuples in non-empty slots, and the remove method is appropriately overridden to throw an UnsupportedOperationException. The hasNext and next methods delegate to the underlying tuple_iterator, which is based on the ArrayList containing non-empty tuples.

Exercise 5

Implementation of HeapFile.java

```
testcompile:

runtest:
    [junit] Running simpledb.HeapFileReadTest
    [junit] Testsuite: simpledb.HeapFileReadTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.147 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.147 sec
    [junit]
    [junit] Testcase: getTupleDesc took 0.08 sec
    [junit] Testcase: numPages took 0.006 sec
    [junit] Testcase: readPage took 0.013 sec
    [junit] Testcase: getId took 0.008 sec
    [junit] Testcase: testIterator0asic took 0.012 sec
    [junit] Testcase: testIteratorClose took 0.013 sec

BUILD SUCCESSFUL
Total time: 2 seconds
```

We implemented methods in HeapFile.java to manage a collection of tuples in a disk without BufferPool calls. We calculate file offsets to read pages from disk without BufferPool calls. We Implemented **HeapFile.iterator()** to iterate through tuples using **BufferPool.getPage()** and avoided loading entire tables into memory on open() to pass the HeapFileReadTest. **readPage** method calculates the correct offset in the file to read a page from disk, avoiding direct BufferPool calls. **writePage** method writes a page to disk, also avoiding direct BufferPool calls. The **insertTuple** method inserts tuples into pages, handling cases where pages are full and creating new pages if necessary. The iterator method returns an iterator for iterating through tuples in the HeapFile. The **HeapFileIterator** class implements the DbFileIterator interface, handling the iteration process by retrieving tuples from pages using the BufferPool.

Exercise 6

Implementation of SeqScan.java

```
testcompile:

runsystest:
    [junit] Running simpledb.systemtest.ScanTest
    [junit] Testsuite: simpledb.systemtest.ScanTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.627 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.627 sec
    [junit]
    [junit] Testcase: testTupleDesc took 0.088 sec
    [junit] Testcase: testCache took 0.186 sec
    [junit] Testcase: testSmall took 0.336 sec
    [junit] Testcase: testRewind took 0.005 sec

BUILD SUCCESSFUL
```

We implemented the following methods to allow for sequential scanning of tuples from the specified table, providing methods to access, iterate over, and manipulate the tuples efficiently. The constructor Initializes the **SeqScan** with the specified transaction ID, table ID, and table alias. It retrieves the database file associated with the table ID and initializes a **DbFileIterator** to iterate over tuples. **getTableName()**: Returns the actual name of the table in the catalog of the database. **getAlias():** Returns the alias of the table. **reset():** Resets the table ID and table alias of the operator. **getTupleDesc():** Returns the TupleDesc with field names prefixed with the table alias, which is useful when joining tables with fields of the same name. **hasNext()**: Checks if there are more tuples to be read. next(): Retrieves the next tuple. rewind(): Rewinds the DbFileIterator to the beginning.