Introduction to Minibase
Database Implementation
CS 487/587

Minibase is a miniature relational DBMS. In these assignments, you will implement simplified versions of different layers of Minibase, without support for concurrency control or recovery, in Java.

These assignments are largely taken from Raghu Ramakrishnan of the University of Wisconsin. We have also used an updated version of the code written by Chris Mayfield and Professor Walif Aref of Purdue University. Phrases in the documentation such as "I have written" are edits by PSU Professor Len Shapiro. We thank all for their development of and work on this code.

You will be given parts of the Minibase code, in Java, and asked to fill in other parts of it. You may spend 90% of your time understanding the given code and 10% of your time writing new code.

Guide to Programming
You will need a Java package and probably will want to use an IDE such as eclipse, available from www.eclipse.org. You may use whatever development environment you chose.

Error Protocol
This version of Minibase makes use of Java's built-in runtime exceptions. In general, only two exceptions are necessary:

- IllegalArgumentException: Throw this when a method argument is invalid, for example if the user attempts to select a record that doesn't exist.
- IllegalStateException: Throw this if the database reaches an undefined state, for example if the user attempts to get the next record after completing a scan.

Required error checking is already included in the comments (i.e. "@throws").
Note that some methods throw exceptions simply because they call other methods that throw them (i.e. BufMgr.newPage() -> BufMgr.pinPage() -> etc).

Grading
For grading purposes, the assignments will be demonstrated to the professor - each demo will be about 5 minutes long and will consist on the tests given in the assignment and on other tests that are not distributed as part of the assignment. You will also be asked to show select pieces of your code. We expect your code to be readable, understandable and use good programming practice. Points will be deducted if we are unable to read/understand your code and for serious inefficiencies. Source code will be turned in, but we will not do a detailed grading of the source code.

Grade breakdown:

80% Passes Tests in Assignments
20% Quality of code: Selected code review for code structure, algorithms, correctness, and readability.

Suggested Programming Practices

- Meaningful variable names
- White space between different logical sections of code
- Comments for methods, complex sections of code, and non-obvious declarations
- Methods approximately no more than one page long
- Efficiency is considered, e.g. not using an integer if a Boolean is appropriate, not doing a sequential search or a disk I/O unless necessary

What to hand in

Your code will be graded during a demonstration time period. Source code will be emailed to the professor (github links are ok also). The submitted source code should include files you have created/written and any other files you have changed. Please include your name in the name of the file submitted.