Testing

We strongly encourage testing your code yourself. The given tests for this project are not comprehensive tests: it is possible to write incorrect code that passes them all (but not get full score).

Things that you might consider testing for include: anything that we specify in the comments or in this document that a method should do that you don't see a test already testing for, and any edge cases that you can think of. Think of what valid inputs might break your code and cause it not to perform as intended, and add a test to make sure things are working. We will **not** be testing behavior on invalid inputs (`null` objects, negative buffer sizes or buffers too small to perform a join, invalid queries, etc...). You can handle these inputs however you want, or not at all.

To help you get started, here is one case that is *not* in the given tests (and will be included in the hidden tests): joining an empty table with another table should result in an iterator that returns no records (`hasNext()` should return false immediately).

To add a unit test, open up the appropriate test file and simply add a new method to the file with a `@Test` annotation, for example:

```
@Test
public void testEmptyBNLJ() {
    // your test code here
}
```

Many test classes have some setup code done for you already: take a look at other tests in the file for an idea of how to write the test code. For example, the SNLJ tests in TestNestedLoopJoin can be used as a template for your own BNLJ, Sort, and SMJ tests.

Previous
Part 2: Query Optimization

Next
Submitting the Assignment

Last updated 5 months ago