**Unit Testing Within the DAO**

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behaviour or state.

The majority of the testing was undertaken within the data access object (DAO). This was due to the high chance for error when coding SQL statements. Our method for unit testing the DAO was:

1. To create a main within the class
2. Create an object with dummy data
3. Then call the method that would perform some action with that dummy data.

Isolate the Error

1. Observe the exception that the statement threw In the console
2. If that didn’t help copy the System.out.println(query) and pasting it into SQLite’s execute box and seeing if the error generated there gave more information
3. Next using the information given by the SQlite error to adjust our statement and start steps 1-3 over again

9 times out of 10 the above steps worked well to isolate problems in our code and helped to assert a certain behaviour or state.

**Step 1,2 and 3**

public static void main(String args[]){

Book book = new Book(11,"somebook“,"James","2015-11-05",45,11);

BookDao dao = new BookDao();

dao.update(book);

}

**Step 4**

System.out.println(QUERY)

QUERY = Update Book set bookID='11', booktitle='somebook', bookAuthor='James', DateAdded='2015-11-05' bookPrice=45.0 where bookID=11

SQLException: syntax error.

**Step 5**

The console exception did not provide enough information. Therefore copy cosole statement into SQLite directly

Update Book set bookID='11', booktitle='somebook', bookAuthor='James', DateAdded='2015-11-05'

SQLException: Surrounding Bookprice syntax error.

**Step 6**

Found error missed coma between 05’ and bookPrice.

Made changes and success

**Unit Testing Within The Frame**

Once we had tested the method within the Dao class we would go on to run it within the frame class . Within this frame class we would again go though the above steps 1-6.

Public class book frame{

book.setbookTitle("somebook");

book.setbookAuthor("James");

book.setdateAdded("2015-11-05");

book.setbookID(1234);

book.setbookPrice(45);

book.setsupplierID(11);

dao.update(book);

}

**White Box Testing**

Finally replace the dummy data with the methods which get the text field information. Also experiment with different numbers and values within GUI

bookPrice = Double.*parseDouble(informationTextField[4].getText());*

bookSuppID = Integer.*parseInt(informationTextField[5].getText());*

bookID = Integer.*parseInt(informationTextField[0].getText());*

book.setbookTitle(informationTextField[1].getText());

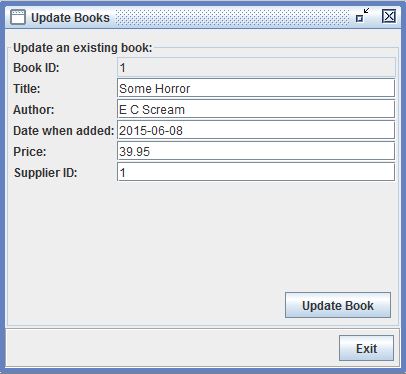
book.setbookAuthor(informationTextField[2].getText());

book.setdateAdded(informationTextField[3].getText());

book.setbookID(bookID);

book.setbookPrice(bookPrice);

book.setsupplierID(bookSuppID);



**Testing summery**

It should be noted that the testing we did for the update book function was not the only test case we did for the project. We recreated these steps with every method we used which accessed the database (15+ methods). We also ran similar test cases which followed a variation on the above testing format but were not related to DAO.

These other tests included things such as: Dialogue boxes would appear when buttons were pressed, conditions like user cannot get more than 6 books out at a time and many others. We kept a few of these extra test cases which can be viewed (commentated out) within the code.

However as you can imagine with so many test cases we found ourselves tripping over commentated out test cases and therefore opted to delete some test cases in order to make the code more readable. Therefore the uncommented out examples you see in the code do not fully exemplify the exhaustive testing we accomplished while involved in the project.