Practical session 12

This work should be completed before the next lecture.

Task 1: Unit testing in your customer application

Copy your NetBeans project from Task 1 of Week 11.

Write JUnit tests to ensure the following:

- Singleton
    - Two calls to `getInstance()` return the same object
- Factory
    - An object of the correct class is returned by the factory method for each value (including invalid values) of the code
- Builder
    - The builder returns a built object with the correct values

Portfolio requirements:

- The NetBeans project for this completed task

Task 2: Unit testing the Lecture11Demo

Download from Blackboard the `Lecture11Demo-Complete` NetBeans project.

Write JUnit tests to ensure the following:

- Copy
    - After creation, state has the correct value
    - After borrowing, state has the correct value
    - After returning, state has the correct value
    - Borrowing a copy twice without having a return operation between throws the correct exception
    - Returning a copy twice without having a borrow operation between throws the correct exception
- Book
    - The constructor throws the correct exception with the correct message when title, author and ISBN are all blank (i.e. the empty string, “”)
    - When there are multiple copies of a book, `findCopy()` returns the correct object
    - When a copy does not exist, trying to find it throws the correct exception
    - Can add five copies
    - Cannot add more than five copies
    - `addCopy()` returns `true` and `false` appropriately

Portfolio requirements:

- The NetBeans project for this completed task

Task 3: Write a test plan

Given the following problem description, write the requirements, test cases and test plan.

Problem

A simple calculator class is needed that permits the user to add, subtract, multiply, and divide decimal point values. It operates on two operands at a time. It also maintains a single memory to which a value can be added (i.e. increasing the current value in memory). The value in memory can be retrieved and also reset to zero. The calculator determines the result of applying the currently selected operator to the current values of the two operands. The result can be retrieved. The calculator can be cleared which resets all values to zero, and the operator to space.

Portfolio requirements:

- A Word document containing the requirements, test cases and test plan

Task 4: Implement a test plan

Create a NetBeans project, and implement your test plan from Task 3 as a set of JUnit tests.

Portfolio requirements:

- The NetBeans project for this completed task

Task 5: Implement an application that satisfies JUnit tests

In your NetBeans project from Task 4, write a Java application that passes all your JUnit tests.

Portfolio requirements:

- The NetBeans project for this completed task