CSC207 Lab 7 — JUnit Testing

1 Getting credit for this lab

Due date: Monday 9 March by 9:00pm

To earn the lab mark, complete the lab individually on your own time and submit:

- your JUnit test suite (`FlightDataRecorderTest.java`), and
- a plaintext file named `bugs.txt` that briefly describes the bugs that you found.

2 Overview

This week, you are going to implement a JUnit test suite.

3 Class `FlightDataRecorder`

Commercial aircraft are required to contain a flight data recorder (FDR), colloquially known as a "black box", for recording data such as aircraft altitude and velocity. An FDR can store up to 25 hours' worth of data; to allow continuous recording, FDRs record on a loop; this means that for a recorder that can store 25 hours' worth of data, at the beginning of the 26th (51st, 76th, etc.) hour of operation, the recorder is overwriting the data from the 1st (26th, 51st, etc.) hour of operations. The FDR code in `FlightDataRecorder` you will be testing in this lab only keeps the last 10 entries of data.

4 Log in and get things set up

`s1` drives and `s2` navigates.

1. Update your subversion repository to get the newly created `lab7` directory. This directory contains directories `Lab7`, `src` and `testinglab`, and file `FlightDataRecorder.java`.
2. Start Eclipse and select `lab7` as a workspace to work in today.
3. Create a new Java Project called `Lab7`. You should now be able to view the starter files in Eclipse, in package `testinglab`.

5 Choosing Test Cases

No driver/navigator. Brainstorm with your lab partner.

- View the Javadocs posted on the course website for the `FlightDataRecorder` class.
- Create a list of test cases by creating a table similar to the one below.

Purpose of test	Expected behaviour
Check the behaviour of `average()` on a new `FlightDataRecorder`.	`average()` returns `0`
...	...

6 Writing unit tests

`s2` drives and `s1` navigates.

- Select `FlightDataRecorder.java` in the Eclipse Package Explorer pane on the left-hand side of the screen.
- From the `File` menu, select `New -> JUnit Test Case`. Note that Eclipse has named the class for you.
- If Eclipse warns you that JUnit 4 is not on your build path, let it add it for you.
- Write at least one test method for each test case in the table. Remember to follow naming conventions by prefixing the name of each test method with `test` and to include the `@Test` annotation before the method definition. Do not forget to document your test methods. **Switch roles halfway through writing your test methods.**
- Click run. If Eclipse asks you which launcher you would like to use, click "Use configuration specific settings" and select the Eclipse JUnit Launcher. Hopefully, you will have discovered some errors. Click on individual test cases in the JUnit pane on the left of the screen for more details about each test case.
- Three of the methods in the provided code contain bugs, for a total of four distinct bugs. One of these bugs will probably take some work to find! You will need to test more than one method, in combination, to identify this bug.

Add all new Java files to the repository. Commit all changes. **Do not include the `bin` folder, the `doc` folder, or Eclipse's "hidden" files!**

Commit `bugs.txt` too!