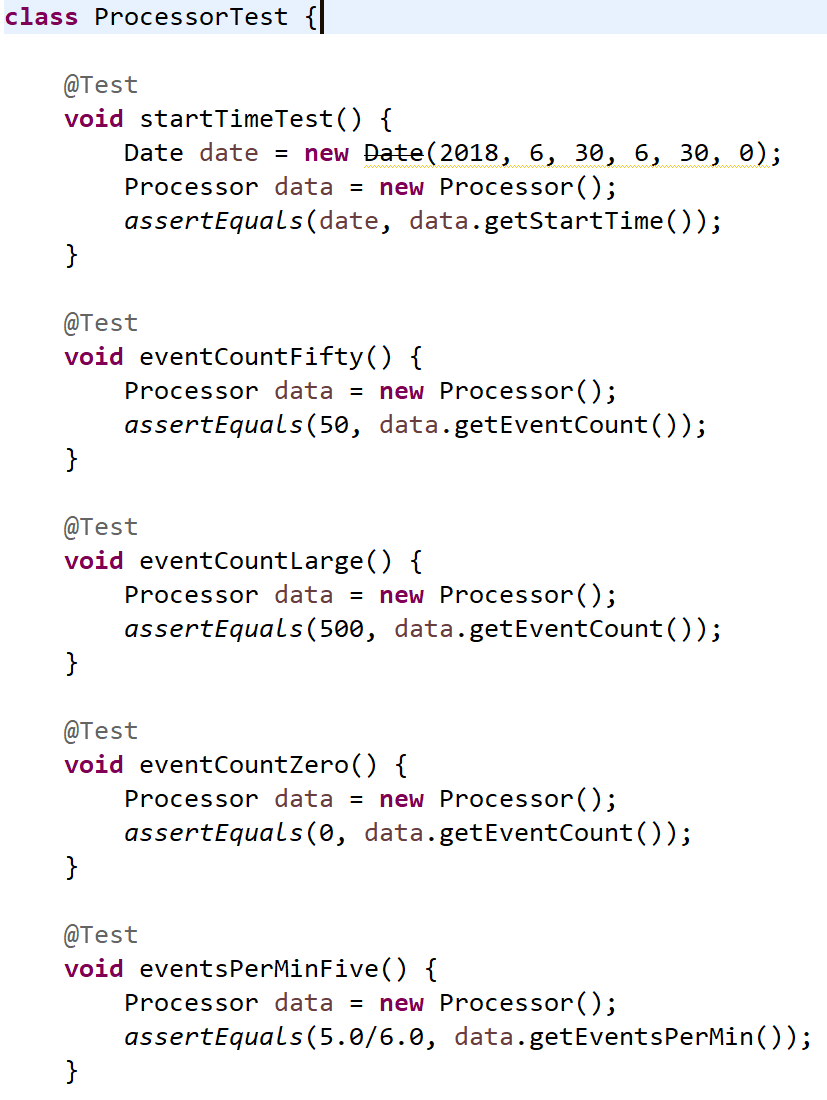
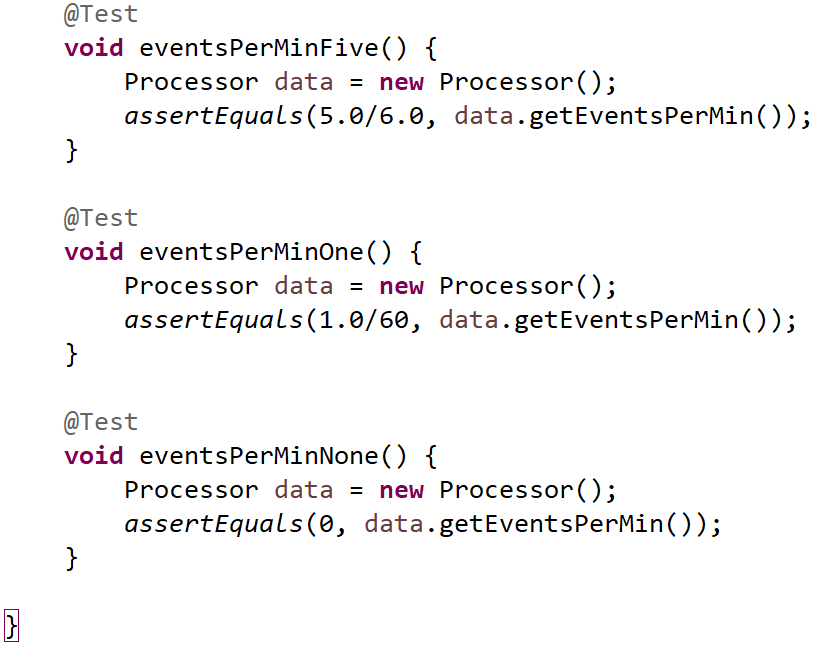
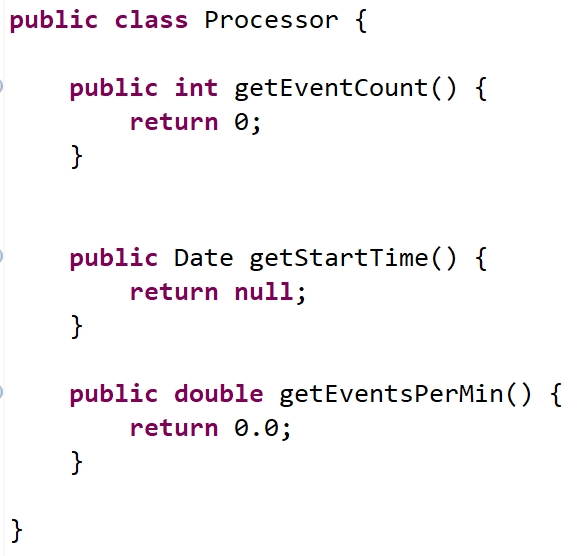
**3-1. Initial Tests**

For the data processing requirement, we will focus on testing the Processor class. This class will store event data as MuonEvents occurring over some time period. Here, we see seven initial test cases that focus on checking if the correct number of events have been stored (by the getEventCount() method) and on calculating the number of events per minute. None of the test code is complete save for the expected values.

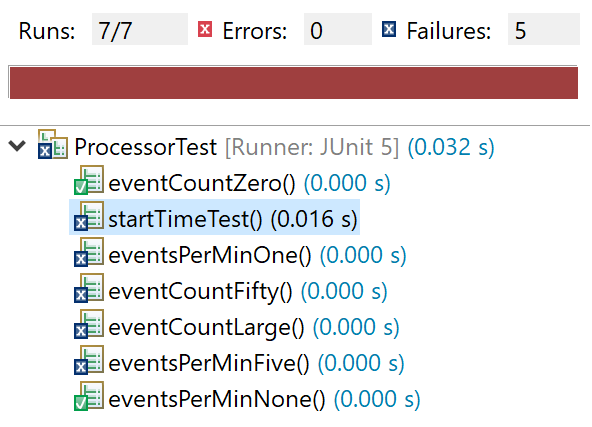




To compile the test code, a basic skeleton of the Processor class was created.



When the tests were run, all but two of the tests failed; the only tests that passed already expected “0” as a result.

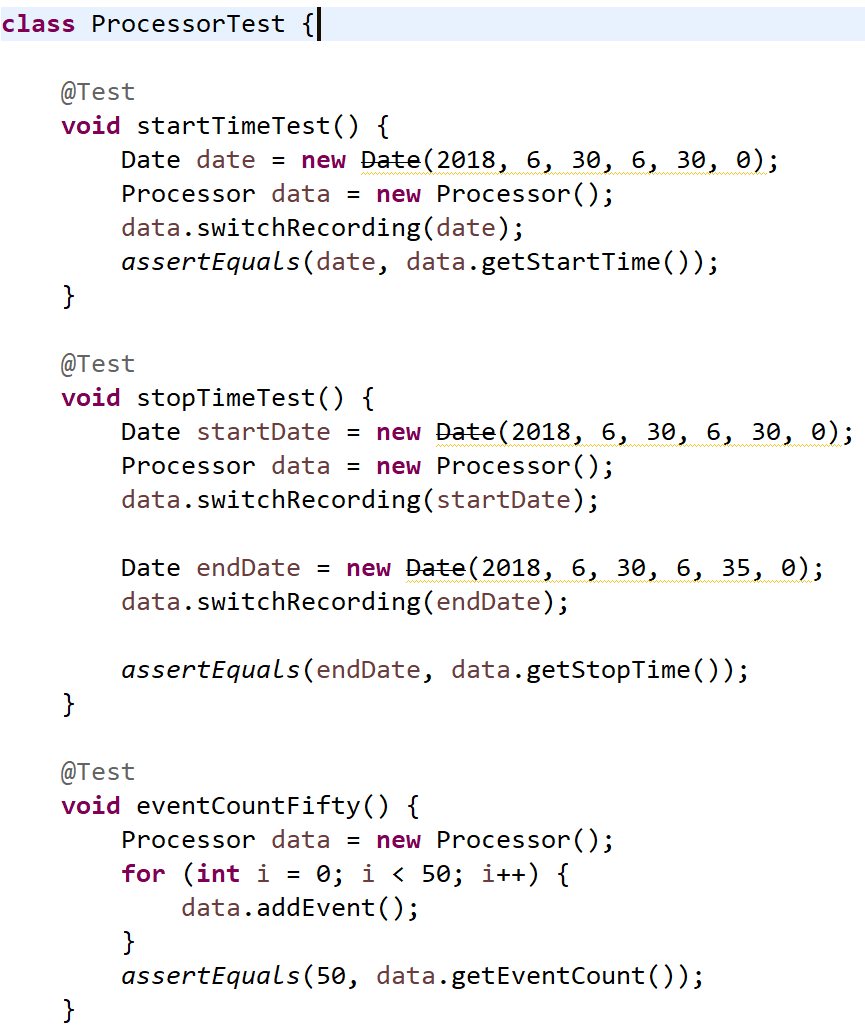


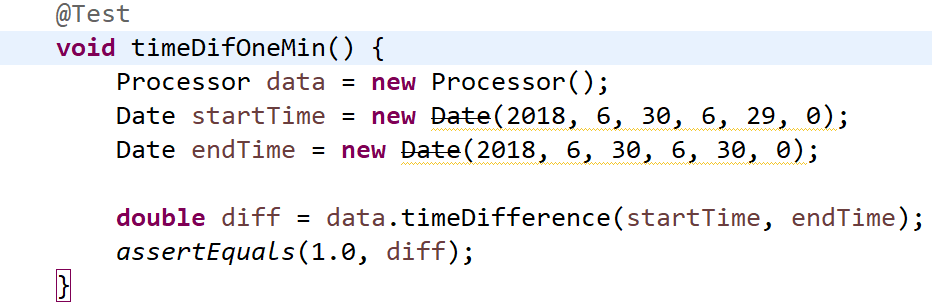
**3.2** **– Tests with Partial Implementation**

While working on the implementation, we realized that we needed more data and methods to flesh out the Processor class, resulting in an increased number of test cases. For example, to calculate events per minute, we needed to save the timestamps when data collection began and ended to know the length of time spent recording data. This functionality is performed by the new switchRecording method.

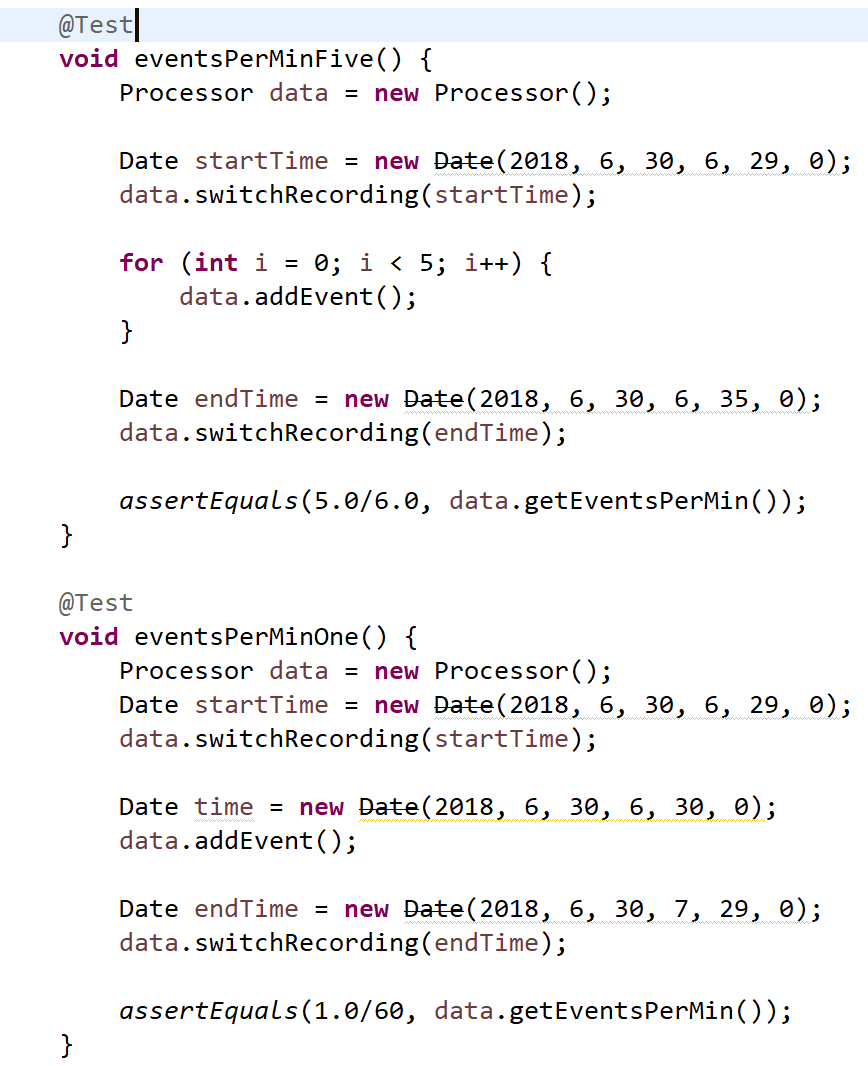
Although the full implementation of this requirement will use real muon event data from a connected muon detector, the test cases created fake data to simulate how real events would be saved by the Processor.

The following screenshots show some of the test cases after implementation:

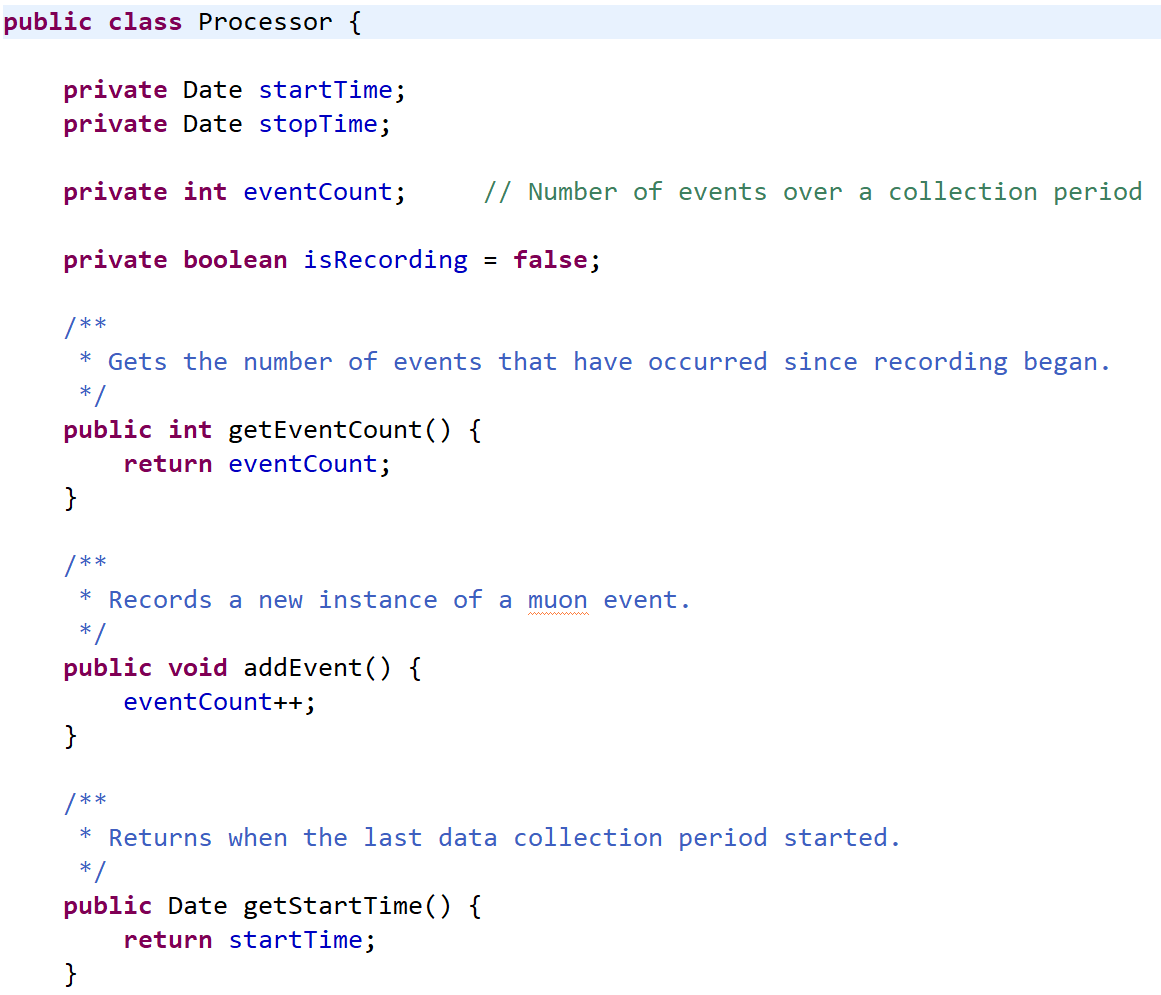


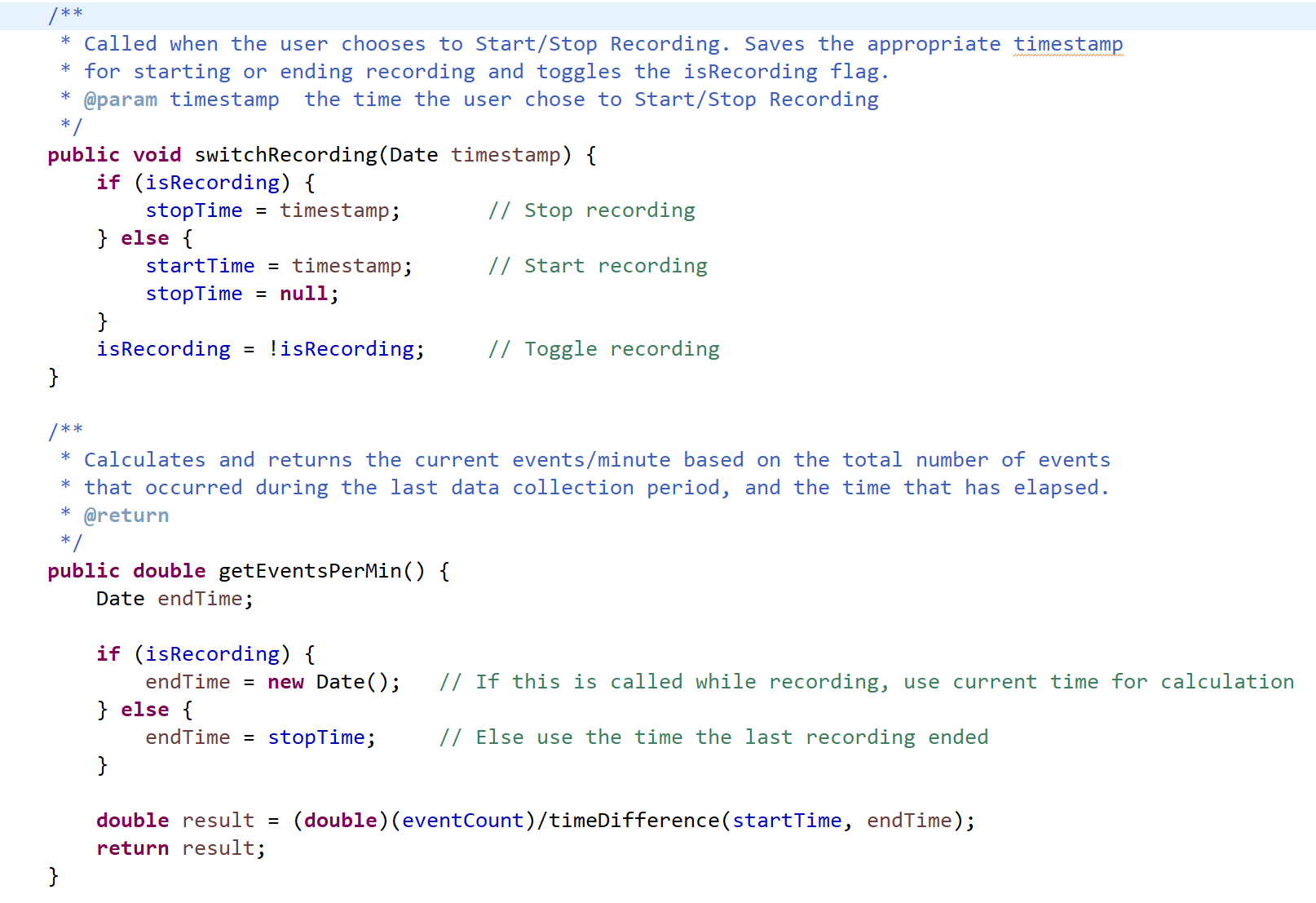


The new timeDifference method was created to calculate the difference between two timestamps in minutes. This is used in conjunction with the getEventsPerMin method.



The Processor class now has partially complete methods. However, it does not yet store instances of each muon event.





Now there is enough code to pass all the test cases:

