Bug Log

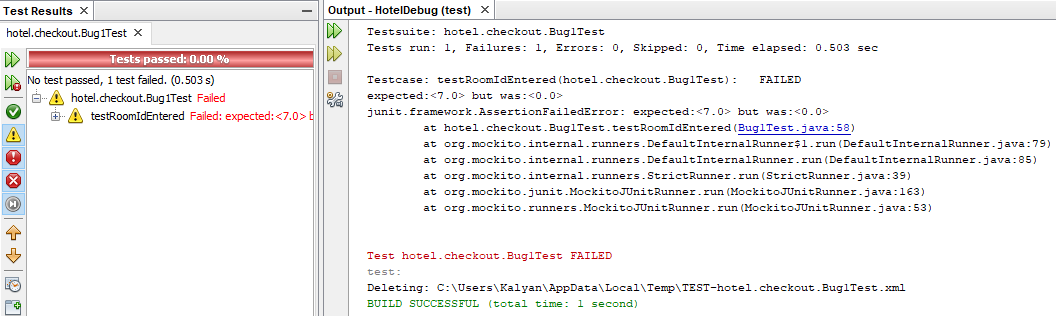
Bug 1: Service Charge $0.00 when Checking Out

9/10/18

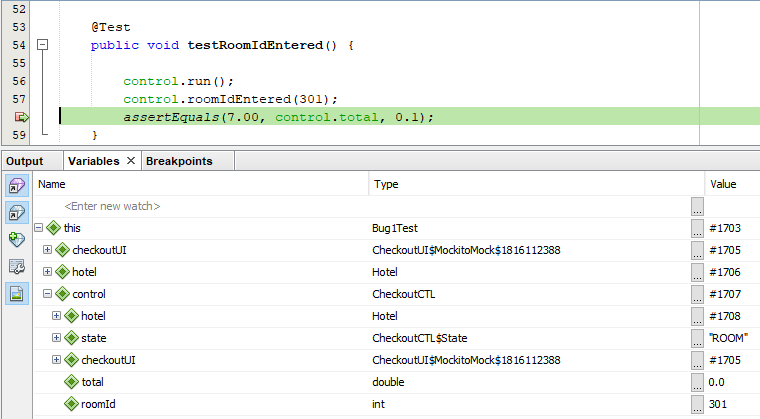
Since the bug occurs when checking out, after the user has entered the room ID, the bug should be able to be replicated by calling the roomIdEntered() method of the checkoutCTL class.

Test method: testRoomIdEntered()

Output from Bug1Test.java demonstrating the bug:



Output from debugger showing control.total is an infected value:



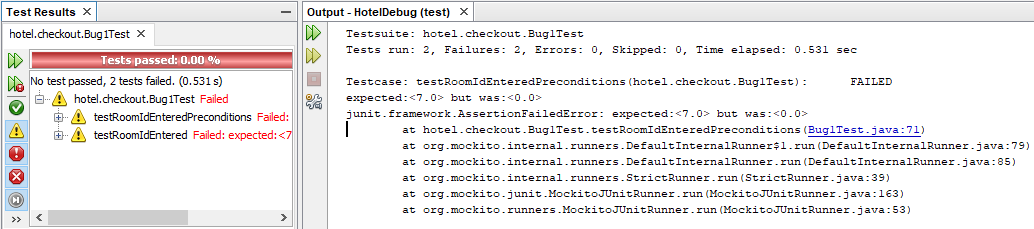
Hypothesis 1: The bug occurs before roomIdEntered() is called

Since the point of failure is in the roomIdEntered() method, the first test should demonstrate whether there is an infected value prior to calling this method.

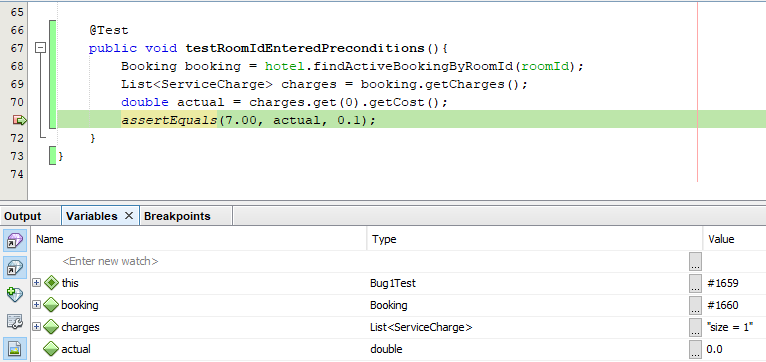
Test method: testRoomIdEnteredPreconditions()

Results: The hypothesis was proven correct, as the service charge was found to be an infected value before roomIdEntered() was called.

Output from Bug1Test.java:



Output from debugger showing that booking.charges contains an infected value:



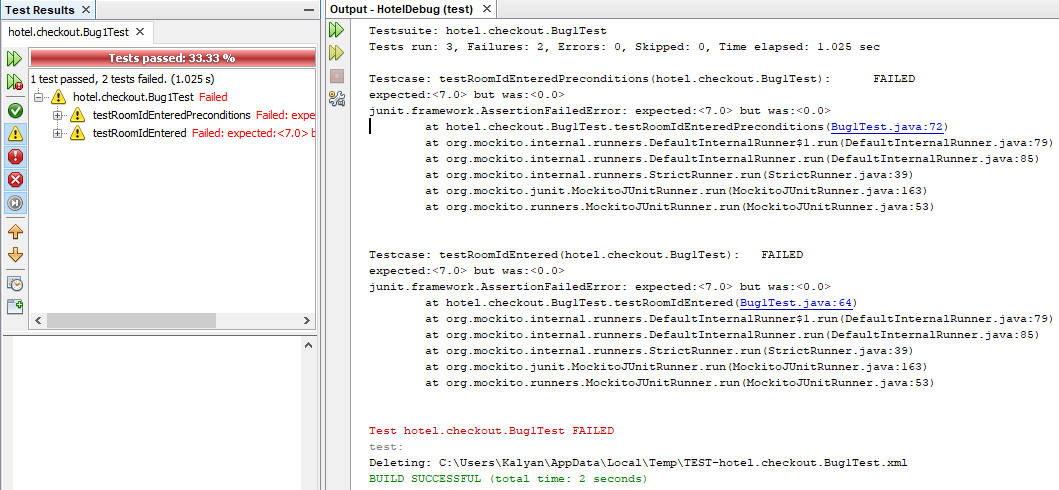
Hypothesis 2: The bug occurs in the constructor of the ServiceCharge class

Since the infected value is the cost field of a ServiceCharge object, the first dependence is the ServiceCharge() constructor which sets that value.

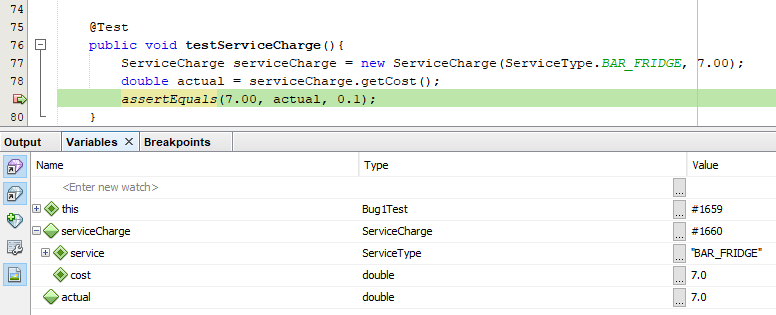
Test method: testServiceCharge()

Results: The test succeeded, showing that the bug is not in the constructor of the ServiceCharge class, thus disproving the hypothesis.

Output from Bug1Test.java:



Output from debugger showing no infected values:



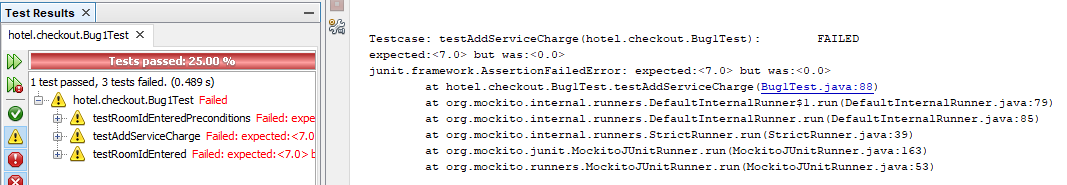
Hypothesis 3: The bug occurs in the addServiceCharge() method of the Booking class

This method, which creates the ServiceCharge using the constructor, is next in the chain of dependences.

Test method: testAddServiceCharge()

Results: The value “cost”, which was provided directly in the test, became infected after calling Booking.addServiceCharge(), thereby proving the hypothesis correct.

Output from Bug1Test.java:



Output from debugger showing that booking.charges[1].cost is an infected value:

