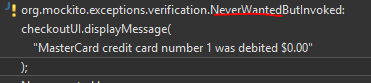
Debugging log

Corey Lovett-Hendrey – ITC205 – Assessment 4

# bug #1: Service charge is $0

## Test #1

### test output before



### Hypothesis

Starting as close to the bug as I can, I assume the value is not being printed correctly.

### Test

First – find where I print the value. I did this by opening the ServiceCharges class, and finding all the calls to “getCost()”. (Ctrl+Shift+G in Eclipse).

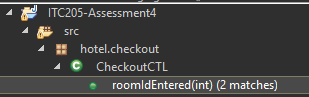


Figure 1: Search window, all calls to getCost()

Then I put a breakpoint on the call to checkoutUI.displayMessage()

I run through the program, ensuring I place a service charge on the room, till I hit the breakpoint.

Here I notice that the ‘charges’ collection contains a 0!

### Result

The printing seems to work fine, however the charges collection contains garbage data (a 0 in this case, which might suggest an initialized int in Javas case – as java never fills an integer with actual garbage)

## test #2

### hypothesis

Knowing the charges collection contains incorrect data, I also notice it is created by a booking.getCharges() method. I might assume this method is returning an incorrect value.

### test

Heading into the getCharges() function, I can see it simply returns the bookings service charge list.

### result

Though the return value *is incorrect*, the method is simply returning an already invalid list! The problem lies before here.

## test #3

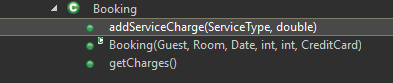
### hypothesis

The ‘charges’ inside Booking are added incorrectly

### test

First, I find where the charges are added to Booking.

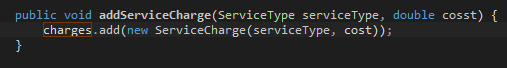
Scrolling to the top of the class I can see charges is a member field of booking, I can search for all references to it. (Ctrl + shift + G in Eclipse)



We already know getCharges() only returns the charges. (Note: Never assume the method name describes what the function *actually* does, I know this returns charges *because we’ve checked it* in a previous test). Booking() is the constructor for Booking, which is where I just called this search from. So that leaves the aptly named “addServiceCharge()”. I will check there.

Checking the function, I see it creates a new service charge and adds it to the charges list. I add a breakpoint to that line, and run the program, making sure to add a service charge to the room.

As soon as I launch the program, I see it wants to add a charge with a cost of 7.0, however I see the variables are named incorrectly! Maybe this is our bug.



The service charge is created with “cost” while the method signature references “cosst”.

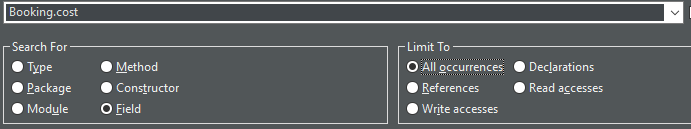
I check these values in the expression window.



This looks like a problem, however, before just changing the value of cosst to cost – I need to understand where the variable “cost” is coming from, to ensure I do not break any other systems.

Highlight cost shows me all occurrences of cost in this class. I notice it is a member field of the incorrect type. I search for all occurrences and notice it is only ever used by the addServiceCharge method. And that method does not need it! Here I really have two options, delete the member variable and rename cosst to cost. Or just do the renaming and let the local cost hide the member cost.

Let’s double check that I am correct in saying cost is not referenced anywhere else. I can use the power of eclipse to perform a Java search for all occurrences of the field.



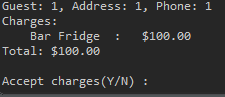
Again, this shows that the field is only referenced by the method that doesn’t need it. I am going to remove it from the code completely at this point and rename cosst to cost.

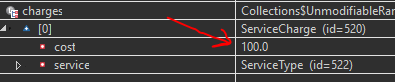
I rerun the application to test its functionality. Here, you would want to run all automated and UAT tests to make sure nothing broke – even after I was so careful!

### result

Re-testing the program – I find the problem to be resolved!



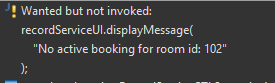




# bug #2: able to charge a checked-out room

The first step in fixing this bug was creating a scenario test that *fails* properly. So, I created a test that runs through the whole program, books a room, checks out then attempts to charge the room again. This test fails if the program successfully charges the room.

### test output before



## test #1

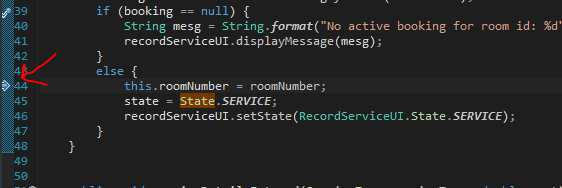
### hypothesis

Starting as close to the bug as possible before working back, I assume the “Add Service Charge” function is acting incorrectly

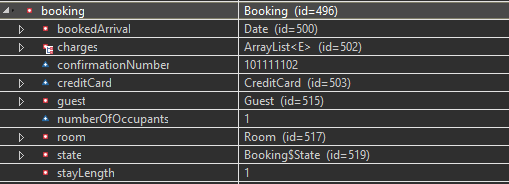
### test

To find where this originates, I find where RecordService starts. (I go from Main -> recordService -> RecordServiceCTL and see it starts with roomNumberEntered.

Reading the function, I see that as long as the Booking is not null it will add a service charge.



We can step through and verify that booking is indeed not null, therefore the service charge gets added.



Here we can see it is not null, so the service stays where it should be deleted.

### result

It appears the booking is never removed after being checked out.

## test #2

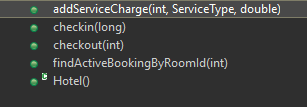
### hypothesis

The booking is not removed correctly from active bookings

### test

Booking is set by “findActiveBookingByRoomId()”, so I venture in there.

I see this just returns a booking from the Map “activeBookingsByID”, so I search for all occurrences of this to see how it is set.



Here we see that it is called by a bunch of functions. I work my way down the list.

addServiceCharge() simply retrieves the booking.

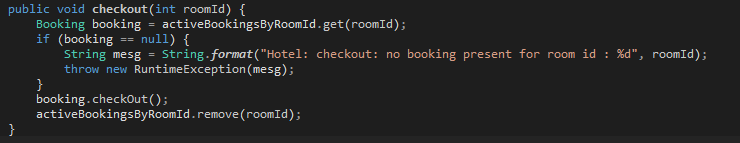
Checkin creates a new booking.

Checkout retrieves the booking

FindActiveBookingByRoomID also just retrieves the booking.

It is never removed!

I go to checkout and remove it there.



I run the test, and it passes.

### result

The test now passes and the user is no longer able to charge a checked-out room!





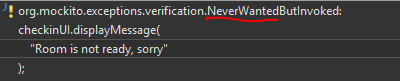




# Bug #3: Checking IN twice displays incorrect error

Checking in to a room that has been checked in should display the “Booking (id) has already been checked in. However, it incorrectly displays “Room is not ready, sorry”.

### test output before



## TEST#1

### hypothesis

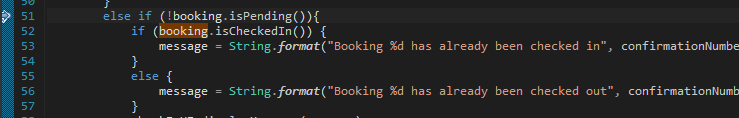
Start at the bug, I hypothesize the function to Checkin is incorrect.

### Test

I follow main through to “confirmationNumberEntered”. This is where the messages are displayed, and as close to the bug as we can get before starting to move backwards.

Here we can see the control flow that leads to all the different error messages. Performing a “static review” of the code shows that the error message that we want should be shown if the booking is not pending, and it is checked in.

I set a break point and step through to this point.



I then use the expressions window to check the return of the calls to booking.



The states are incorrect! The booking should not be pending, and it should be checked in.

### result

We have another state issue. This will lead us to a more accurate hypothesis.

## test #2

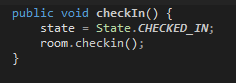
### hypothesis

The checked in state is not being set correctly in booking

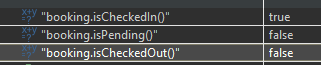
### test

So I go into “booking.isPending()” first. It just returns whether the state is “PENDING”. I search for all occurrences of this and see that the class is set to pending in the constructor. I then check “isCheckedIn()” and search for all occurrences of CHECKED\_IN. It is never set!

I set it in the checkin() method, and rerun the test. It passes.



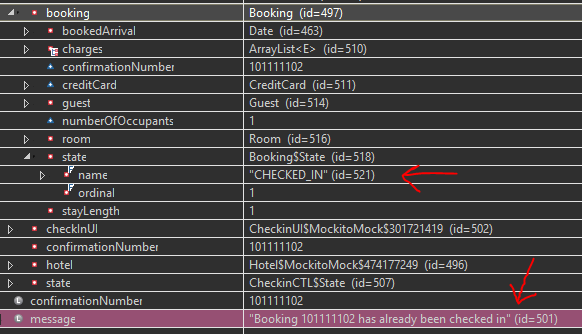
Stepping through the code we can now see the states are correct.



### result

The bug is fixed. The tests pass, and the states are set correctly.



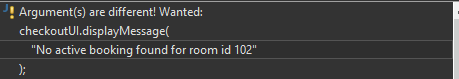




# bug #4: CHECKING INTO A CHECKED-OUT ROOM

When a user attempts to check into a room that has already been checked out, the error displayed is “Room is already *checked* ***in****”* instead of already checked **out**.

### Test output before



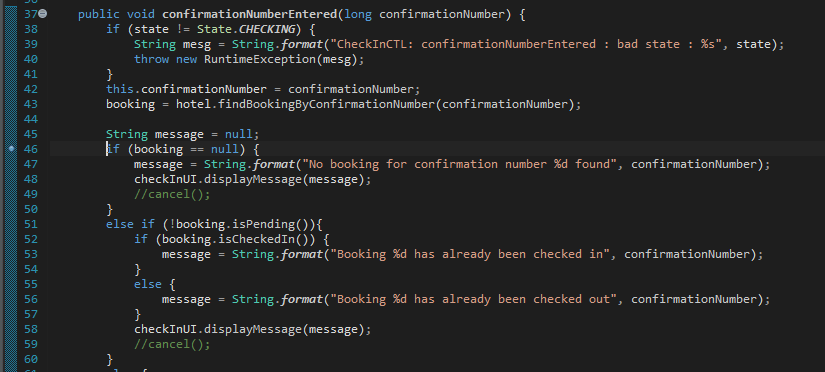
## test #1

### hypothesis

The message being displayed by check in is printing incorrectly.

### test

Inspect where the message is printed when checking in.



Placing a breakpoint at the start of the main control flow (line 46) I step through the code when running the test in debug mode.

I use the expressions window again to inspect the return value of booking.isCheckedIn().

This should be false! As we can see, stepping through because this is true we get the incorrect error message.

### result

The booking state is incorrect. It returns CHECKED\_IN when it should be CHECKED\_OUT

## test #3

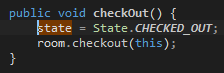
### hypothesis

The booking state is being set incorrectly

### test

I go into the ‘isCheckedIn’ on Booking. I see that it just returns the current state. I do a search for all occurrences of the “CHECKED\_OUT” state and see that it is not being set anywhere!

I look at the “checkout” function. It just calls room.checkout. I add the state change.



I run the test again, and it passes.

### result

The test passes and the correct error message is displayed.



