

Simplification:

Record output:

The code below is the code that is provided with the Hotel java files. This class has all the necessary dummy data that is needed for the testing of all the test cases. Thus, this class is used for the dummy data. It is done by instantiating the Hotel object named hotel to HotelHelper.loadHotel() with the try catch exception so that there will not be any problem if there is null exception or invalid data.

After declaring and instantiating this class, the required method are tested.

```
public class HotelHelper {

    public static Hotel loadHotel() throws Exception {

        //SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        //Date date = format.parse("01-01-2001");
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
        Date date = format.parse("01-01-0001");
        cal.setTime(date);

        Hotel hotel = new Hotel();
        // hotel.addRoom(RoomType.SINGLE, 101);
        // hotel.addRoom(RoomType.DOUBLE, 201);
        // hotel.addRoom(RoomType.TWIN_SHARE, 301);

        Guest guest = new Guest("Fred", "Nurke", 2);
        CreditCard card = new CreditCard(CreditCardType.VISA, 2, 2);

        Room room = hotel.findAvailableRoom(RoomType.DOUBLE, date, 1);
        long confNo = hotel.book(room, guest, date, 1, 2, card);
        Booking booking = hotel.findBookingByConfirmationNumber(confNo);
        hotel.checkin(confNo);
        booking.addServiceCharge(ServiceType.ROOM_SERVICE, 7.00);

        IOUtils.trace("HotelHelper");
        for (RoomType rt : RoomType.values()) {
            Map<Integer, Room> rooms = hotel.roomsByType.get(rt);
            for (Integer id : rooms.keySet()) {
                IOUtils.outputln(rooms.get(id));
            }
        }
    }
}
```

Here, for the automated testing to reproduce the bug two methods are tested.

1. roomIdEntered(int roomId) of CheckoutCTL
 2. serviceDetailsEntered(ServiceType serviceType, double cost) of RecordServiceCTL
- a) CheckoutCTL.roomIdEntered
- This method is checked to for the service charge. When the user checks out the total service charge should be given according to the service they have used.

But here according to the bug report, no matter how much the user uses the service, the total charge will be \$0.00 .

The following code does the exam same.

```
public void testRoomIdEntered() {
    System.out.println("roomIdEntered");

    //declaring the hotel object
    Hotel hotel = null;

    //trying the try catch so that any exception that is occurred will be
    captured.
    try {
        //instanciating the hotel object
        hotel = HotelHelper.loadHotel();
    } catch (Exception ex) {
        Logger.getLogger(CheckoutCTLTest.class.getName()).log(Level.SEVERE,
null, ex);
    }

    //the room ID used in hotelhelper class is 301. So, that is the room
    ID that is being used.
    int roomId =201;

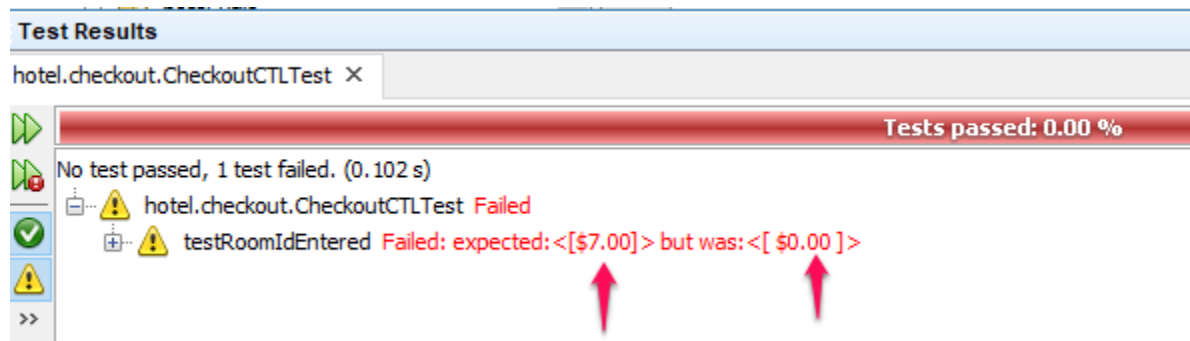
    //creating a checkoutCTL object and then passing the hotel object as
    an argument.
    CheckoutCTL instance = new CheckoutCTL(hotel);

    //calling the method to check the roomId that is being entered.
    instance.roomIdEntered(roomId);

    //the output is then converted to string and then stored to the
    variable output.
    String output = outContent.toString();

    //using the assertEquals to check the value of the string
    //Now comparing the exact value that is expected to the result shown
    //here substring is used to get the total dollar value from all of the
    result of the output
    assertEquals("$7.00",output.substring(output.lastIndexOf(":")+1,
output.length()-1 ));
}
```

The code above produces the result below:



Here, the code did not pass because the service charge should be \$7.00 but the charge was \$0.00 as the bug report mentioned. This service charge was applied to room 201.

1. serviceDetailsEntered(ServiceType serviceType, double cost) of RecordServiceCTL

```

/**
 * Test of creditDetailsEntered method, of class CheckoutCTL.
 * /*Checking the possiblility of charging a room for service after the guest has checked out
 */
@Test
Run Test | Debug Test
public void testCheckOutAndChargeRoom() {
    System.out.println("Checkout from the room and then add service charge");
    Hotel hotel = null;

    //using the roomId that is used in the hotelhelper class
    int roomId= 301;
    try {
        hotel = HotelHelper.loadHotel();
    } catch (Exception ex) {
        Logger.getLogger(CheckoutCTLTest.class.getName()).log(Level.SEVERE, null, ex);
    }

    //checking out of the hotel
    hotel.checkout(roomId);

    System.out.println("Checking serviceDetailsEntered");
    //adding the room service charge of 20
    ServiceType serviceType = ServiceType.ROOM_SERVICE;
    double cost = 20.0;

    //creating the instance of the record service
    RecordServiceCTL instance = new RecordServiceCTL(hotel);
    instance.roomNumberEntered(roomId);
    //adding the service name and charge to the record service
    instance.serviceDetailsEntered(serviceType, cost);
    String output = outContent.toString();

```

```

//creating the instance of the record service
RecordServiceCTL instance = new RecordServiceCTL(hotel);
instance.roomNumberEntered(roomId);
//adding the service name and charge to the record service
instance.serviceDetailsEntered(serviceType, cost);
String output = outContent.toString();






//here the output should be no active booking found
//But this will add service charge to the room ID even if this room has checked out
assertEquals("No active booking for room id: 201",output.substring(output.lastIndexOf("Entered")

```

Output:

Test Results

hotel.checkout.CheckoutCTLTest X



Tests passed: 100.00 %

The test passed. (0.078 s)

- hotel.checkout.CheckoutCTLTest passed
 - testCheckOutAndChargeRoom passed (0.016 s)

