**Black-Box Test Cases for Reservation Constructor and Getter Methods:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Test Description | Selected Inputs  custID, roomType, StartDate, endDate | Expected Result | Actual Result | Pass/Fail |
| 1 | Reservation ID Uniqueness | R1 = 1, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025”  R2 = 7, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025” | Two different reservations are created, so two different UUID will be tested. | Actual:271e8a8c-ce7b-45da-9ed5-ba8698ae2e0c does not equal Expected:88ddb5ff-5dbc-481a-95f6-7cdfea2aaca3 PASS | Pass |
| 2 | Reservation Creation Date | R1 = 1, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025”  newDateObject = new Date() | Today’s date is Feb 17, 2025, therefore r1 and the newDateObject have the same date since the Reservation object is created in the same day | Actual:Mon Feb 17 16:33:09 CET 2025 does not equal  Expected:Mon Feb 17 16:33:09 CET 2025 FAIL | Fail  Due to millisecond difference between reservation creation date and date object creation |
| 3 | Guest ID Uniqueness | R1 = 1, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025”  R2 = 7, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025” | Two different guests have the same room type and dates in their reservations, but their IDs are different | Actual:1 does not equal Expected:7 PASS | Pass |
| 4 | Start Date and End Date Format | R2 = 7, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025” | Both dates are different, therefore the test will pass | Actual:Feb 18, 2025 does not equal Expected:Feb 21, 2025 PASS | Pass |
| 5 | Rooms with Similar Types | R1 = 1, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025”  R2 = 7, “RoomWBath”, “Feb 18, 2025”, “Feb 21, 2025” | Both rooms have similar types, therefore the test will pass | Actual:RoomWBath equals Expected:RoomWBath PASS | Pass |

**Unit Test Code:**

A screen shot of a computer code

AI-generated content may be incorrect.

Results:A screenshot of a computer

AI-generated content may be incorrect.

**Approach, Steps, and Rationale:**

When creating the unit test for the constructor and getters method, I followed the example given in the instructions. The example tested the Reservation constructor, getReservationID(), and getReservationDate() methods. Once I had an idea of how to use the methods in the Assert class, I started testing the getter method. I tested the getGuestID() method by asserting if both reservations had different guest IDs. After testing the getGuestID() method I moved on to the getReservationStartDate() and getReservationEndDate() methods. I wanted to test if the strings were not equal. Lastly, I tested the getRoomType().

**Black-Box Test Cases for Reservation Setter and Getter Methods:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Test Description | Selected Inputs  custID, roomType, StartDate, endDate | Expected Result | Actual Result | Pass/Fail |
| 1 | Setting a new guest ID | R1 = 7, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  R1.setGuestID(3) | R1’s guest ID will be changed from 7 to 3 | Actual:3 does not equal Expected:7 PASS | Pass |
| 2 | Setting a new room type | R1 = 7, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  R1.setRoom(“NormalRoom”) | R1’s room type will be changed from RoomWBath to NormalRoom | Actual:NormalRoom does not equalExpected:RoomWBath PASS | Pass |
| 3 | Updating reservation creation date | R1 = 7, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  r1.setGuestID(3); r1.setRoom("NormalRoom"); r1.setReservationStartDate("Mar 01, 2025"); r1.setReservationEndDate("Mar 25, 2025"); | Setting R1’s information to new dates, room type, and guest ID will change the creation of the reservation date to a new one | Actual:Mon Feb 17 16:35:38 CET 2025 equals Expected:Mon Feb 17 16:35:38 CET 2025 FAIL | Fail  The creation date of the reservation date stays the same |
| 4 | Setting a new reservation start date | R1 = 7, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  r1.setReservationStartDate("Mar 01, 2025"); | R1’s reservationStartDate will be changed from “Feb 16, 2025” to “Mar 01, 2025” | Actual:Feb 16, 2025 equals Expected:Feb 16, 2025 FAIL | Fail  Due to setReservationStartDate() not changing the date. |
| 5 | Setting a new reservation end date | R1 = 7, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  r1.setReservationEndDate("Mar 25, 2025"); | R1’s reservationEndDate will be changed from “Feb 19, 2025” to “Mar 25, 2025” | Actual:Mar 25, 2025 does not equal Expected:Feb 19, 2025 PASS | Pass |

**Unit Test Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

Results:

A computer screen shot of a computer

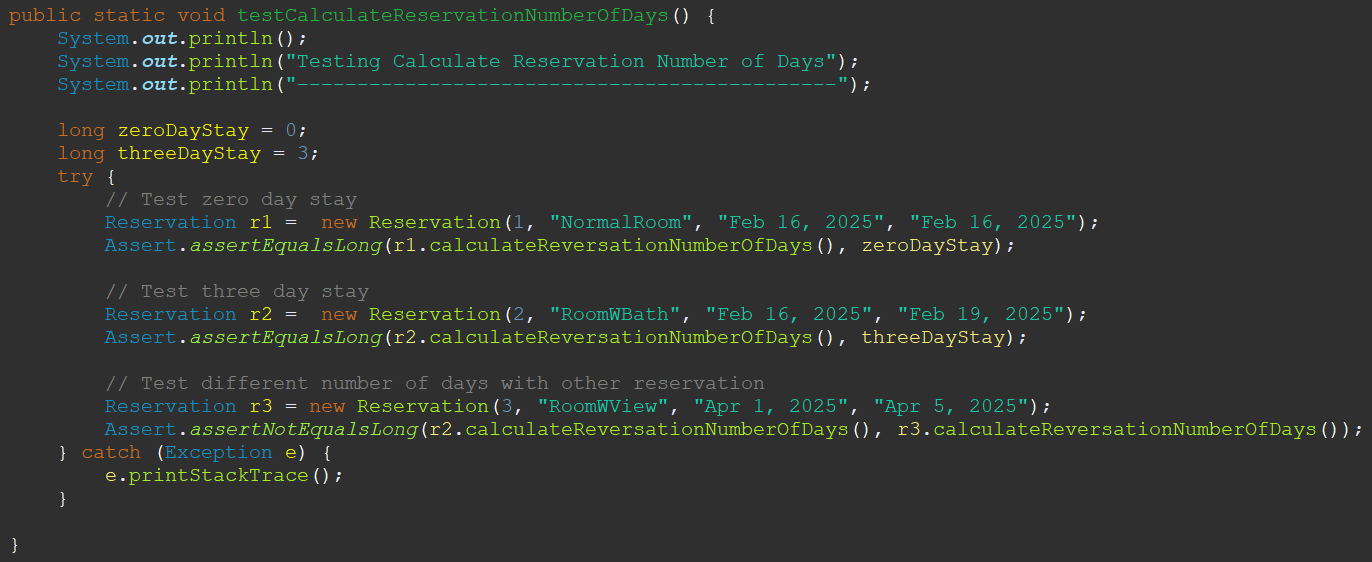
AI-generated content may be incorrect.

**Approach, Steps, and Rationale:**

When creating the unit test for the setters and getters method, I wanted to first create a new Reservation object and use the getter methods to store the current information into variables such as previousRoomType and previousGuestID. Once I stored the current information into variables, I started to test the setter methods to change the information of the Reservation object. After using every setter method the Reservation object has, I wanted to test and prove that the setters did in fact change the information. I checked each getter method with the respective variable containing the old information.

**Black-Box Test Cases Reservation calculateReservationNumberOfDays() Method:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Test Description | Selected Inputs  custID, roomType, StartDate, endDate | Expected Result | Actual Result | Pass/Fail |
| 1 | Reservation for one day (not one night) | R1 = 1, “NormalRoom”, “Feb 16, 2025”, “Feb 16, 2025”  zeroDayStay = 0; | The r1.calculateReversationNumberOfDays() method returns 0. | Actual:0 equals Expected:0 PASS | Pass |
| 2 | Reservation for three days | R2 = 2, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  threeDayStay = 3; | The r2.calculateReversationNumberOfDays() method returns 3. | Actual:3 equals Expected:3 PASS | Fail |
| 3 | Check if two reservation have the same number of days | R2 = 2, “RoomWBath”, “Feb 16, 2025”, “Feb 19, 2025”  R3 = 3, “RoomWView”, “Apr 1, 2025”, “Apr 5, 2025” | The r2.calculateReversationNumberOfDays() method returns 3.  The r3.calculateReversationNumberOfDays() method returns 4. | Actual:1 does not equal Expected:7 PASS | Pass |

**Unit Test Code:** 

Results:

A screen shot of a black background

AI-generated content may be incorrect.

**Approach, Steps, and Rationale:**

Since the Reservation class assumes that the constructor always has valid dates, I wanted to test if the program knows that a reservation is only for one day (not one night). I created a new reservation object with the start date: Feb 16, 2025, and end date: Feb 16, 2025. I then tested the calculateReservationNumberOfDays() method to see if it’ll return zero. The next test I wanted to conduct was checking a normal three day stay. I followed the same format as the first but created a new object with the dates “Feb 16, 2025” – “Feb 19, 2025.” Lastly, I wanted to test a situation where there are two reservations with different lengths of stays. This was done by using the previous Reservation with a new Reservation.

**Black-Box Test Cases for Reservation calculateReservationBillAmount() Method:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Test Description | Selected Inputs  custID, roomType, StartDate, endDate | Expected Result | Actual Result | Pass/Fail |
| 1 | Calculating normal room reservation bill for three days | R4 = 4, “NormalRoom”, “Feb 17, 2025”, “Feb 20, 2025”  normalRoomPrice = 125.0; | R4.calculateReservationBillAmount() returns $375 | Actual:360.0  does not equal Expected:375.0 FAIL | Fail  Due to method not properly calculating NormalRoom Reservation bill |
| 2 | Calculating normal room reservation bill for 14 days | R4.setReservationEndDate(“Mar 03, 2025”)  normalRoomPrice = 125.0; | R4.calculateReservationBillAmount() returns $1750 | Actual:1680.0  does not equal Expected:1750.0 FAIL | Fail  Due to method not properly calculating NormalRoom Reservation bill |
| 3 | Calculating normal room reservation bill for one month | R4.setReservationEndDate(“Mar 19, 2025”)  normalRoomPrice = 125.0; | R4.calculateReservationBillAmount() returns $3750 | Actual:3600.0  does not equal Expected:3750.0 FAIL | Fail  Due to method not properly calculating NormalRoom Reservation bill |
| 4 | Calculating room w/ view reservation bill for three days | R5 = 5, “RoomWView”, “Feb 17, 2025”, “Feb 20, 2025”  roomWViewPrice = 175.0; | R5.calculateReservationBillAmount() returns $525 | Actual:525.0 equals Expected:525.0  PASS | Pass |
| 5 | Calculating room w/ view reservation bill for 14 days | R5.setReservationEndDate(“Mar 03, 2025”)  roomWViewPrice = 175.0; | R5.calculateReservationBillAmount() returns $2450 | Actual:2450.0 equals Expected:2450.0  PASS | Pass |
| 6 | Calculating room w/ view reservation bill for one month | R5.setReservationEndDate(“Mar 19, 2025”)  roomWViewPrice = 175.0; | R5.calculateReservationBillAmount() returns $5250 | Actual:5250.0 equals Expected:5250.0  PASS | Pass |
| 7 | Calculating room w/ bath reservation bill for three days | R6 = 6, “RoomWBath”, “Feb 17, 2025”, “Feb 20, 2025”  roomWBathPrice = 200.0; | R6.calculateReservationBillAmount() returns $600 | Actual:0.0  does not equal Expected:600.0 FAIL | Fail  Due to method not properly calculating Room w/ bath Reservation bill |
| 8 | Calculating room w/ bath reservation bill for 14 days | R6.setReservationEndDate(“Mar 03, 2025”)  roomWBathPrice = 200.0; | R6.calculateReservationBillAmount() returns $2800 | Actual:6000.0  does not equal Expected:2800.0 FAIL | Fail  Due to method not properly calculating Room w/ bath Reservation bill |
| 9 | Calculating room w/ bath reservation bill for one month | R6.setReservationEndDate(“Mar 19, 2025”)  roomWBathPrice = 200.0; | R6.calculateReservationBillAmount() returns $6000 | Actual:60000.0  does not equal Expected:6000.0 FAIL | Fail  Due to method not properly calculating Room w/ bath Reservation bill |

**Unit Test Code:** A screen shot of a computer program

AI-generated content may be incorrect.

Results:

A screenshot of a computer

AI-generated content may be incorrect.

**Approach, Steps, and Rationale:**

For this unit test, I wanted to test if the method calculated the bill properly for three situations, a stay for three days, 14 days, and one month, each with different room types. The idea behind the test is to test the normal amount of stay, a long stay, and a longer stay. This was to check if the actual bill generated, scaled properly as the number of days the guest stays increases.

**Lessons Learned:**

Overall, this assignment gave me the opportunity to conduct a proper black-box unit test. Being assigned a .jar file where I cannot see the contents, really drilled the point of how a black-box test works. However, creating tests for the Reservation class was somewhat challenging. The specification document for the Reservation class was helpful and helped guide how I tailored the tests for each section of the class. I was able to look at the coding assignment from a different perspective, and that gave me a better understanding of software testing.