Name: Joseph Baskin  
Date: 11/22/2024  
Week: 6 - Black-box Unit Test the Reservation Class of a Small Bed & Breakfast Reservation System

Embed here a copy of your complete Java unit test source code (e.g., TestReservation.java):

import java.text.\*;  
import java.util.\*;  
  
public class TestReservation {  
 private static String *datePattern* = "MMM dd, yyyy";  
 private static SimpleDateFormat *sdf* = new SimpleDateFormat(*datePattern*);  
  
 public static void main(String[] args) throws Exception {  
 *testConstructorAndGetters*();  
 }  
  
 public static void testConstructorAndGetters() throws Exception {  
 System.*out*.println();  
 System.*out*.println("Testing Constructor and Getters:");  
 System.*out*.println("--------------------------------");  
  
 Reservation r = new Reservation(1, "RoomWBath", "JUN 16, 2022", "JUN 19, 2022");  
 Reservation r2 = new Reservation(7, "RoomWBath", "Jun 16, 2022", "Jun 19, 2022");  
 Assert.assertNotEqualsUUID(r.getReservationID(), r2.getReservationID());  
 Assert.assertEqualsDate(r.getReservationDate(), new Date());  
 Assert.assertEqualsString(r.getRoomType(), r2.getRoomType());  
 Assert.assertNotEqualsInt(r.getGuestID(), r2.getGuestID());  
 Assert.assertEqualsDate(*sdf*.parse(r.getReservationStartDate()), *sdf*.parse(r2.getReservationStartDate()));  
 Assert.assertNotEqualsDate(*sdf*.parse(r.getReservationEndDate()), *sdf*.parse(r2.getReservationEndDate()));  
  
 System.*out*.println();  
 System.*out*.print("Before r.setGuestID: " + r.getGuestID() + " | ");  
 r.setGuestID(3);  
 System.*out*.println("After r.setGuestID: " + r.getGuestID());  
 System.*out*.print("Before r2.setRoom: " + r2.getRoomType() + " | ");  
 r2.setRoom("NormalRoom");  
 System.*out*.println("After r2.setRoom: " + r2.getRoomType());  
 System.*out*.print("Before r2.setReservationStartDate: " + r2.getReservationStartDate() + " | ");  
 r2.setReservationStartDate("DEC 05, 2024");  
 System.*out*.println("After r2.setReservationStartDate: " + r2.getReservationStartDate());  
 System.*out*.print("Before r2.setReservationEndDate: " + r2.getReservationEndDate() + " | ");  
 r2.setReservationEndDate("Dec 11, 2024");  
 System.*out*.println("After r2.setReservationEndDate: " + r2.getReservationEndDate());  
  
 System.*out*.println();  
 System.*out*.println("Reservation r number of days: " + r.calculateReversationNumberOfDays());  
 System.*out*.println("Reservation r2 number of days: " + r2.calculateReversationNumberOfDays());  
 Assert.assertEqualsLong(r.calculateReversationNumberOfDays(), r2.calculateReversationNumberOfDays());  
 Assert.assertNotEqualsLong(r.calculateReversationNumberOfDays(), r2.calculateReversationNumberOfDays());  
  
 System.*out*.println();  
 System.*out*.println("Reservation r bill amount: $" + r.calculateReservationBillAmount());  
 System.*out*.println("Reservation r2 bill amount: $" + r2.calculateReservationBillAmount());  
 Assert.assertEqualsDouble(r.calculateReservationBillAmount(), (r.calculateReversationNumberOfDays() \* 200.00));  
 Assert.assertEqualsDouble(r2.calculateReservationBillAmount(), (r2.calculateReversationNumberOfDays() \* 125.00));  
 }  
}

-----------------------------------------------------------------------------------------------------------------------------------

Rubric Criteria:  
Create black-box test cases to test the constructor and the getters methods of the Reservation class 8%  
Your Response:

The example that is shown in the assignment already had 2 black-box tests, so I created 4 additional to test different inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Selected Inputs (guestID, roomType, startDate, endDate) | Expected Result | Actual Result | Pass | Fail |
| 1 | 1, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022”  &  7, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022” | roomType | roomType equal each other | Pass |
| 2 | 1, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022”  &  7, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022” | guestID | guesID does not equal each other | Pass |
| 3 | 1, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022”  &  7, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022” | startDate | startDates equal each other | Pass |
| 4 | 1, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022”  &  7, “RoomWBath”, “Jun 16, 2022”, “Jun 19, 2022” | endDate | endDates do not equal each other | Fail as both end dates are the same. |

Rubric Criteria:  
Execute, using w6.jar, unit tests for the constructor and the getters method of the Reservation class. Document the unit tests code and results via screenshots 10%  
Your Response:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Rubric Criteria:  
Explain approach, steps, and rationale of the test cases and unit tests of testing the constructor and the getters method of the Reservation class 5%  
Your Response:

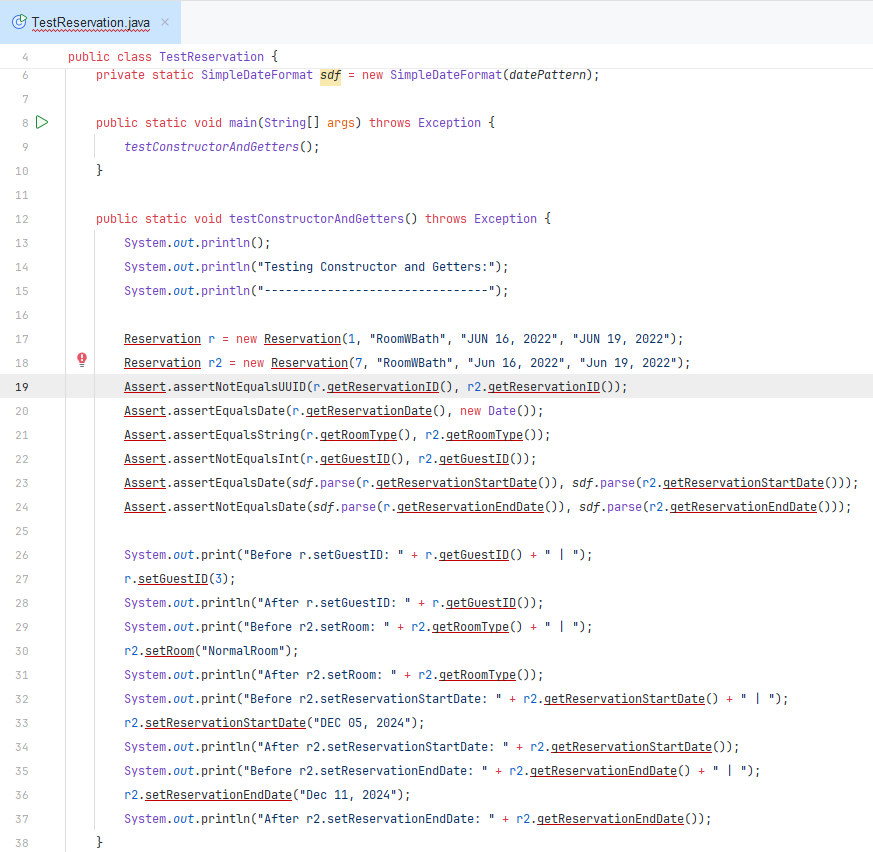
Since the example showed two test cases in the assignment, I decided to make 4 new test cases. I wanted to test the roomType and guestID because in a real program, these would be valuable for the operators to ensure no duplicate reservations. I also wanted to test the startDate and endDate for the reservation to make sure the equals and not equals works. This would allow for an updateReservation() method to be made to easily update either date, and a notification when creating reservations that one exists for the roomType.

-----------------------------------------------------------------------------------------------------------------------------------

Rubric Criteria:  
Create black-box test cases to test the setters and the getters methods of the Reservation class 8%  
Your Response:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Selected Inputs | Expected Result | Actual Result | Pass | Fail |
| 1 | r.setGuestID(3) | r.guestID updates to 3 from previous value | r.guestID updated to 3 from 1 | PASS |
| 2 | r2.setRoom(“NormalRoom”) | r2.roomType updates to “NormalRoom” | r2.roomType updated to “NormalRoom” | PASS |
| 3 | r2.setReservationStartDate(“DEC 05,2024”) | r2.reservationStartDate updates to “DEC 05, 2024” | r2.reservationStartDate did not update | FAIL |
| 4 | r2.setReservationEndDate(“Dec 11,2024”) | r2.reservatopmEmdDate updates to “Dec 11, 2024” | r2.reservationEndDate updated to “Dec 11, 2024” | PASS |

Rubric Criteria:  
Execute, using w6.jar, unit tests for the setters and the getters method of the Reservation class. Document the unit tests code and results via screenshots 10%  
Your Response:



A screenshot of a computer

Description automatically generated

Rubric Criteria:  
Explain approach, steps, and rationale of the test cases and unit tests of testing the setters and the getters method of the Reservation class 5%  
Your Response:

Needed to test each class variable (guestID, roomType, reservationStartDate, and reservationEndDate) in case the owners ever needed to provide updates to reservations, so those became the focal point of my black-box testing. I discovered that there may be an error with the Reservation.setReservationStartDate() updating the wrong variable. In my testing, I noticed that after running my third test and fourth test, the Reservation.reservationEndDate got updated instead of the Start Date for the constructor.

-----------------------------------------------------------------------------------------------------------------------------------

Rubric Criteria:  
Create black-box test cases to test the calculateReservationNumberOfDays() method of the Reservation class 8%  
Your Response:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Selected Inputs | Expected Result | Actual Result | Pass | Fail |
| 1 | r.calculateReservationNumberOfDays() | 3 | 3 | Pass |
| 2 | r2.calculateReservationNumberOfDays() | 7 | 909 | Fail |
| 3 | Assert.assertEqualsLong(r.calculateReversationNumberOfDays(), r2.calculateReversationNumberOfDays()) | Values are not equal | Values are not equal | Pass |
| 4 | Assert.assertNotEqualsLong(r.calculateReversationNumberOfDays(), r2.calculateReversationNumberOfDays()); | Values are not Equal | Values are not Equal | Pass |

Rubric Criteria:  
Execute, using w6.jar, unit tests for the calculateReservationNumberOfDays() method of the Reservation class. Document the unit tests code and results via screenshots 10%  
Your Response:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Rubric Criteria:  
Explain approach, steps, and rationale of the test cases and unit tests of testing the calculateReservationNumberOfDays() method of the Reservation class 5%  
Your Response:

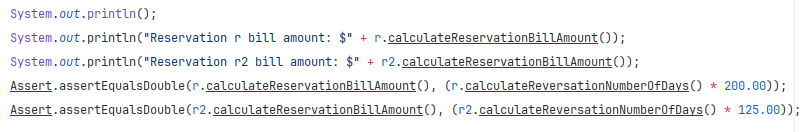
Short of creating new reservations and/or messing with the reservation Start and End dates of either reservation, there wasn’t really 4 black-box tests that I could see running, so I decided to try using the Assert class to check if the number of days were equal and not equal.

-----------------------------------------------------------------------------------------------------------------------------------

Rubric Criteria:  
Create black-box test cases to test the calculateReservationBillAmount() method of the Reservation class 8%  
Your Response:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Selected Inputs | Expected Result | Actual Result | Pass | Fail |
| 1 | r.calculateReservationBillAmount() | 600 | 6000 | Fail |
| 2 | r2.calculateReservationBillAmount() | 875 | 109080 | Fail |
| 3 | Assert.assertEqualsLong(r.calculateReversationBillAmount(), r.calculateReversationNumberOfDays() \* 200.00) | Values are equal | Values are not equal | Fail |
| 4 | Assert.assertNotEqualsLong(r2.calculateReversationBillAmount(), r2.calculateReversationNumberOfDays() \* 125.00); | Values are equal | Values are not Equal | Fail |

Rubric Criteria:  
Execute, using w6.jar, unit tests for the calculateReservationBillAmount() method of the Reservation class. Document the unit tests code and results via screenshots 10%  
Your Response:



A screenshot of a computer

Description automatically generated

Rubric Criteria:  
Explain approach, steps, and rationale of the test cases and unit tests of testing the calculateReservationBillAmount() method of the Reservation class 5%  
Your Response:

Same rationale as the previous method uses and testing. I used the values from Specifications of the Reservation class to get the amount of each room and multiply by the calculated number of days for the actual results but calculated everything myself for what I expected (7 days from Dec 5 – Dec 11, vs 909 because of the potential error in the code).

-----------------------------------------------------------------------------------------------------------------------------------

Rubric Criteria:  
Reflect on the learning experience and lessons learned 8%  
Your Response:

It wasn’t clear that I didn’t have to develop the Reservation and Assert classes for this assignment, which I ended up spending several hours on. I also found that not having the source code to review is the most painful way to test any software during development. Certainly not something I would prefer to do.