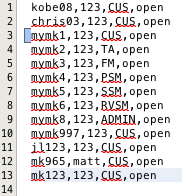
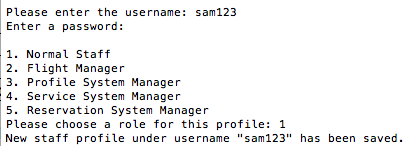
Test Cases (Black Box)

|  |
| --- |
| **Test Case ID:** 1 |
| **Actors Affected:** System Administrator, Normal Staff |
| **Purpose:** To check whether a staff profile is created correctly. |
| **Expected Results:**   1. The database should contain the details entered into the system. 2. The user should be able to login to the system with the same details. 3. The user should be given the correct role. |
| **Output:**   1. The database contains the same details entered into the system 2. The user is able to login to the system with the same details. 3. The user is given the correct role. |
| **Pass/Fail:** Pass |
| **Resultant Changes:** None |
| **Error Messages:** None |

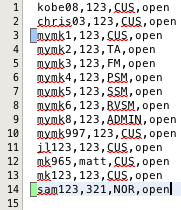
**Database entries before new record:**



**User input into the system:**

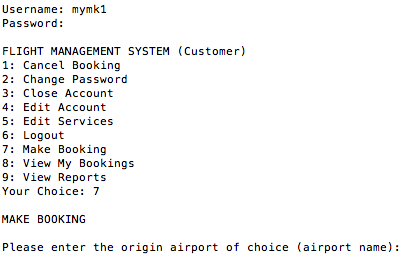


**Database entries after new record:**

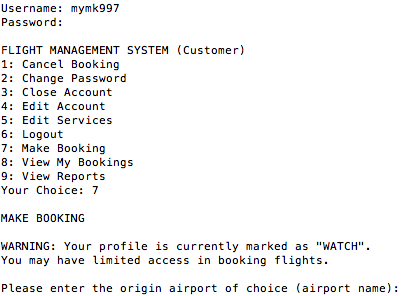


|  |
| --- |
| **Test Case ID:** 2 |
| **Actors Affected:** Customer |
| **Purpose:** To check whether the system responds correctly to the three types of fly status of a customer: “null” (means can fly), “watch” or “no fly”. |
| **Expected Results:**   1. The system should retrieve the correct fly status for the current customer. 2. Based on the retrieved fly status, the system should perform the appropriate actions:    * A customer that is not marked on the no fly list (“null”) is allowed to perform bookings.    * A customer that is marked “watch” is shown a warning message by the system, but is still allowed to perform bookings.    * A customer that is marked “no fly” is not allowed to make a flight booking. |
| **Output:**   1. The system retrieves the correct fly status for the current customer. 2. A customer that is not marked on the no fly list (“null”) was allowed to perform bookings. 3. A customer that is marked “watch” was shown a warning message by the system, but was still allowed to perform bookings 4. A customer that is marked “no fly” was shown a warning message and was not allowed to make a flight booking. |
| **Pass/Fail:** Pass |
| **Resultant Changes:** None |
| **Error Messages:** None |

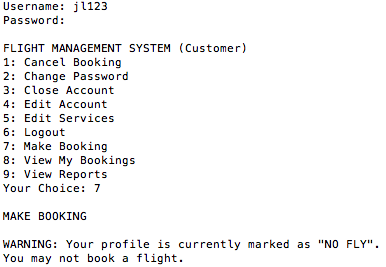
**Customer that can fly:**



**Customer that has “watch” on his record**:



**Customer that has “no fly” on his record:**

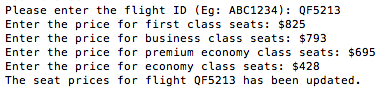


|  |
| --- |
| **Test Case ID:** 3 |
| **Actors Affected:** Reservation System Manager |
| **Purpose:** To check whether the system changes the price of the correct flight selected by the user. |
| **Expected Results:**   1. The system should store the price input from the user in the flight\_schedule.csv file. 2. The system should update the prices of the correct flight. |
| **Output:**   1. The system stored the price input from the user in the flight\_schedule.csv file. 2. The system updated the prices of the correct flight. |
| **Pass/Fail:** Pass |
| **Resultant Changes:** None |
| **Error Messages:** None |

**Database entry before update:**



**User input into the system:**



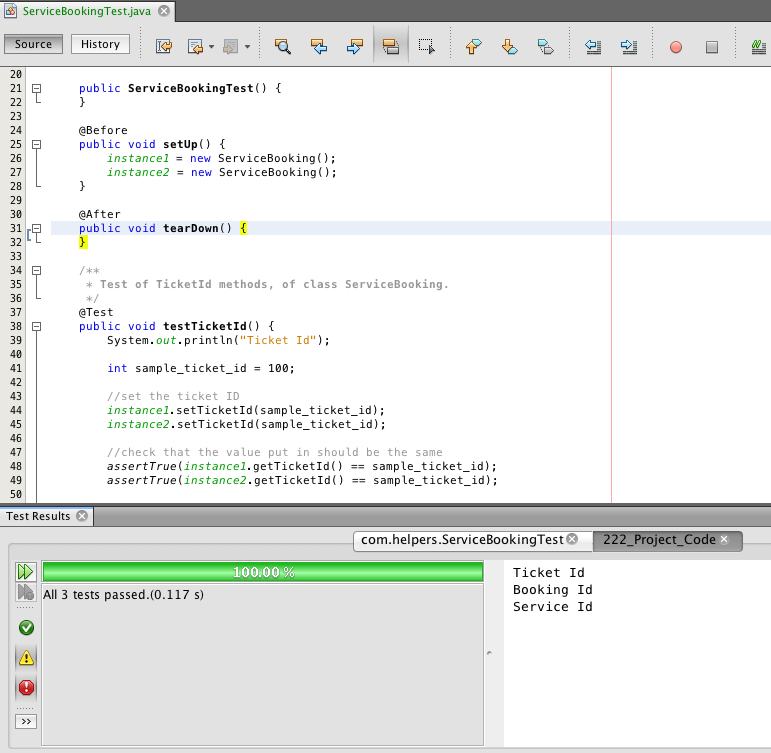
**Database entry after update:**



Test Cases (White Box Unit Testing)

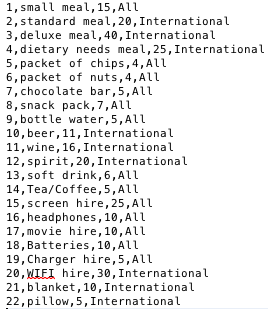
|  |
| --- |
| **Test Case ID:** 4 |
| **Class Tested:** ServiceBooking.java |
| **Tester Class:** ServiceBookingTest.java |
| **Number of Subtests:** 3 |
| **Purpose:** To check whether the get and set methods of ServiceBooking.java are correct and work as expected. |
| **Expected Results:**   1. We are testing the three attributes of the class and the corresponding get and set methods of these attributes:  * bookingId: int * ticketId: int * serviceId: int  1. For each attribute, we will test whether the value stored in the class matches what was passed to it initially. 2. For bookingId:    1. We pass in a value of 5 to the setBookingId method of two instances of the ServiceBooking class.    2. Then we assert that the value returned by the getBookingId method must be the same as the value that was passed in earlier.    3. Lastly we assert that the value returned by the getBookingId method of the first instance must be the same as the value returned by the getBookingId method of the second instance. 3. For ticketId:    1. We pass in a value of 100 to the setTicketId method of two instances of the ServiceBooking class.    2. Then we assert that the value returned by the getTicketId method must be the same as the value that was passed in earlier.    3. Lastly we assert that the value returned by the getTicketId method of the first instance must be the same as the value returned by the getTicketId method of the second instance. 4. For serviceId:    1. We pass in a value of 38 to the setServiceId method of two instances of the ServiceBooking class.    2. Then we assert that the value returned by the getServiceId method must be the same as the value that was passed in earlier.    3. Lastly we assert that the value returned by the getServiceId method of the first instance must be the same as the value returned by the getServiceId method of the second instance. |
| **Output:**   1. The class passed all three subtests. No errors were recorded. |
| **Pass/Fail:** Pass |
| **Resultant Changes:** None |
| **Error Messages:** None |

**Part of the test code with the test results:**



|  |
| --- |
| **Test Case ID:** 5 |
| **Class Tested:** ServiceEntity.java |
| **Tester Class:** ServiceEntity Test.java |
| **Number of Subtests:** 3 |
| **Purpose:** To check whether the database mapping methods of ServiceEntity.java are correct and work as expected. |
| **Expected Results:**   1. We are testing 3 data retrieval methods of the ServiceEntity.java class: 2. getServices(): List<Service> 3. getService(serviceId: int): Service 4. getPrice(serviceId: int): double 5. For item i., we will first get a List of non-international services. Next, we will get a List of all services. Refer to the screenshot of the test data below this table. 6. For item ii., we will create a new Service object with the attributes in the the database entry with service ID of 1. Based on this Service, we will retrieve the same service from the database and compare it for equality against our created Service object. 7. For item iii., we selected item with ID of 12. So the corresponding price for this service is $20. So we assert that this must be true. |
| **Output:**   1. The class passed all three subtests. No errors were recorded. |
| **Pass/Fail:** Pass |
| **Resultant Changes:** None |
| **Error Messages:** None |

**Test data:**



**Part of the test code with the test results:**

