# 3.3 – 2.3.4\_03: Unit Tests Unit Test – Seats Date: 30.03.11 – JUnit – Whitebox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | static Seats screen1 = new Seats("Screen 1");  static Staff stf1 = new Staff("James", "Nightingale", 20, "07123283823", 456, "Manager", "jn1", "lalalala");  static Film flm1 = new Film("Paul", 15);  static FilmShowing fs1 = new FilmShowing(flm1, "27/03/2011", screen1, 6.99);  static Customer cst1 = new Customer("Dave", "Russell", 21, "07533475113");  static Booking booking = new Booking(cst1, stf1, fs1); Seats seats = new Seats(booking); |  |  |  |  |
| 1 | assertNotNull("Booking ID is null", seats.getIntBookingID()); | Pass | Pass | N/A | None taken |
| 2 | assertNotNull("Showing ID is null", seats.getIntShowingID()); | Pass | Pass | N/A | None taken |
| 3 | assertNotNull("Customer ID is null", seats.getIntCustomerID()); | Pass | Pass | N/A | None taken |
| 4 | assertNotNull("Screen is null", screen1.getStrScreen()); | Pass | Pass | N/A | None taken |
| 5 | assertNotNull("Price is null", seats.getDoubleTotalPrice()); | Pass | Pass | N/A | None taken |
| 6 | assertSame("Screens do not match", screen1.getStrScreen(), fs1.getStrScreen()); | Pass | Pass | N/A | None taken |
| 7 | int test;  assertSame("Return error", seats.setIntNumSeats(2), test = seats.setIntNumSeats(2));  assertSame("Number of seats do not match", seats.setIntNumSeats(2),seats.getIntNumSeats()); | Pass | Pass | N/A | None taken |
| 8 | assertNotNull("Number of seats is null", seats.getIntNumSeats()); | Pass | Pass | N/A | None taken |
| 9 | assertTrue("Row is unrecognised", seats.setObjRow(Row.A)); | Pass | Pass | N/A | None taken |
| 10 | seats.setObjRow(Row.A);  assertNotNull(seats.getObjRow()); | Pass | Pass | N/A | None taken |
| 11 | assertTrue("Column is unrecognised", seats.setObjColumn(Column.J)); | Pass | Pass | N/A | None taken |
| 12 | seats.setObjColumn(Column.L);  assertNotNull("Column has not been set", seats.getObjColumn()); | Pass | Pass | N/A | None taken |
| 13 | try {  assertTrue("Adding has failed - Returned false", screen1.addSeat(seats));  } catch (Exception e) {  fail("Adding has failed - Exception caught");  } | Pass | Pass | N/A | None taken |
| 14 | screen1.addSeat(newSeats);  assertNotNull(screen1.toString()); | Pass | Pass | N/A | None taken |
| 15 | assertNotNull("Get date/time failed",seats.getCurrentDateTime()); |  |  |  |  |

# Unit Test – Booking Date: 30.03.2011 – JUnit – Whitebox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | Seats screen1 = new Seats("Screen 1");  Booking booking = new Booking("Bookings");  Staff stf1 = new Staff("James", "Nightingale", 20, "07123283823", 456, "Manager", "jn1", "lalalala");  Film flm1 = new Film("Paul", 15);  FilmShowing fs1 = new FilmShowing(flm1, "27/03/2011", screen1, 6.99);  Customer cst1 = new Customer("Dave", "Russell", 21, "07533475113");  Booking staffBooking = new Booking(stf1);  Booking sncBooking = new Booking(cst1, stf1);  Booking fullBooking = new Booking(cst1, stf1, fs1)  Booking testBooking; |  |  |  |  |
| 1 | assertNotNull(testBooking = new Booking()); | Pass | Pass | N/A | None taken |
| 2 | assertNotNull(staffBooking.toString()); | Pass | Pass | N/A | None taken |
| 3 | assertNotNull(testBooking = new Booking(stf1)); | Pass | Pass | N/A | None taken |
| 4 | assertNotNull(testBooking = new Booking(cst1,stf1)); | Pass | Pass | N/A | None taken |
| 5 | assertNotNull(testBooking = new Booking(cst1, stf1, fs1)); | Pass | Pass | N/A | None taken |
| 6 | assertNotNull(staffBooking.getIntBookingID());  assertNotSame(staffBooking.getIntBookingID(), sncBooking.getIntBookingID()); | Pass | Pass | N/A | None taken |
| 7 | assertNotNull(fullBooking.getDblTotalPrice()); | Pass | Pass | N/A | None taken |
| 8 | assertNotNull(staffBooking.getIntStaffID()); | Pass | Pass | N/A | None taken |
| 9 | assertNotNull(fullBooking.getIntShowingID()); | Pass | Pass | N/A | None taken |
| 10 | assertNotNull(fullBooking.getIntCustomerID()); | Pass | Pass | N/A | None taken |
| 11 | assertNotNull(fullBooking.getIntFilmRating()); | Pass | Pass | N/A | None taken |
| 12 | assertNotNull(fullBooking.getShowingTime()); | Pass | Pass | N/A | None taken |
| 13 | booking.addBooking(fullBooking);  assertNotNull(booking); | Pass | Pass | N/A | None taken |
| 14 | assertNotNull(fullBooking.generateBookingID()); | Pass | Pass | N/A | None taken |
| 15 | assertNotNull(fullBooking.getDateTime()); | Pass | Pass | N/A | None taken |
| 16 | assertNotNull(fullBooking.toString()); | Pass | Pass | N/A | None taken |

# Unit Test – FilmShowing Date: 30.03.2011 – JUnit – Whitebox

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | | | | | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | Seats screen1 = new Seats("Screen 1");  FilmShowing showings = new FilmShowing("Showings");  Film flm1 = new Film("Paul", 15);  FilmShowing testFS;  FilmShowing fs1 = new FilmShowing(flm1, "27/03/2011", screen1, 6.99); | | | | |  |  |  |  |
| 1 | showings.addFilmShowing(fs1);  System.out.println(fs1); | | | | | Pass | Pass | N/A | None taken |
| 2 | assertNotNull(testFS = new FilmShowing(flm1)); | | | | | Pass | Pass | N/A | None taken |
| 3 | assertNotNull(testFS = new FilmShowing("Showings")); | | | | | Pass | Pass | N/A | None taken |
| 4 | assertNotNull(testFS = new FilmShowing(flm1, "21/03/2009", screen1, 4.99)); | | | | | Pass | Pass | N/A | None taken |
| 5 | showings.addFilmShowing(fs1);  showings.showShowings(); | | | | | Pass | Pass | N/A | None taken |
| 6 | assertNotNull(fs1.getIntShowingID()); | | | | | Pass | Pass | N/A | None taken |
| 7 | fs1.setStrDate("21/03/2009"); | Pass | Fail | java.lang.IllegalArgumentException:  Cannot format given Object as a Date | Tasked developer to fix method. | | | | |
| 8 | assertNotNull(fs1.getStrDate()); | | | | | Pass | Pass | N/A | None taken |
| 9 | assertNotNull(fs1.getStrScreen()); | | | | | Pass | Pass | N/A | None taken |
| 10 | fs1.setStrScreen("Screen 2");  System.out.println(fs1.toString());  assertNotNull(fs1.getStrScreen()); | | | | | Pass | Pass | N/A | None taken |
| 11 | assertNotNull(fs1.getDblPrice()); | | | | | Pass | Pass | N/A | None taken |
| 12 | fs1.setIntPrice(7.99);  System.out.println(fs1); | | | | | Pass | Pass | N/A | None taken |
| 13 | showings.addFilmShowing(fs1); | | | | | Pass | Pass | N/A | None taken |
| 14 | showings.showShowings(); | | | | | Pass | Pass | N/A | None taken |
| 15 | showings.addFilmShowing(fs1);  assertNotNull(showings.getByID(2));  System.out.println(showings.getByID(2)); | | | | | Pass | Pass | N/A | None taken |
| 16 | showings.addFilmShowing(fs1);  assertNotNull(showings.getByRating(2));  System.out.println(showings.getByRating(2)); | | | | | Pass | Pass | N/A | None taken |
| 17 | showings.addFilmShowing(fs1);  assertNotNull(showings.getByPrice(2));  System.out.println(showings.getByPrice(2)); | | | | | Pass | Pass | N/A | None taken |
| 18 | testFS = new FilmShowing();  assertNotNull(testFS.generateFSID()); | | | | | Pass | Pass | N/A | None taken |
| 19 | assertNotNull(fs1.getDateTime()); | | | | | Pass | Pass | N/A | None taken |

# Unit Test – Customer Date: 30.03.2011 – JUnit – Whitebox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | Customer customers = new Customer("Customers");  Customer testArray;  Person testCust;  Customer cust1 = new Customer("Dave", "Russell", 21, "01232 123123");  Customer cust2 = new Customer(); |  |  |  |  |
| 1 | assertNotNull(testCust = new Customer()); | Pass | Pass | N/A | None taken |
| 2 | assertNotNull(testCust = new Customer("James", "Nightingale", 20, "01232 123123")); | Pass | Pass | N/A | None taken |
| 3 | assertNotNull(testArray = new Customer("Customers")); | Pass | Pass | N/A | None taken |
| 4 | assertNotNull(cust1.getCustomerID()); | Pass | Pass | N/A | None taken |
| 5 | customers.addCustomer(cust1); | Pass | Pass | N/A | None taken |
| 6 | assertNotNull(cust2.generateCustomerID()); | Pass | Pass | N/A | None taken |
| 7 | System.out.println(cust1); | Pass | Pass | N/A | None taken |

# Unit Test – Staff Date: 30.03.2011 – JUnit – Whitebox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | Staff staff = new Staff("Staff");  Staff testStaff;  Staff stf1 = new Staff("Amanda", "Patterson", 21, "01010 929292", "Head Honcho", "ap1", "ap1"); |  |  |  |  |
| 1 | assertNotNull(stf1.getStrFirstName()); | Pass | Pass | N/A | None taken |
| 2 | stf1.setStrFirstName("Manda");  assertNotNull(stf1.getStrFirstName()); | Pass | Pass | N/A | None taken |
| 3 | assertNotNull(stf1.getStrLastName()); | Pass | Pass | N/A | None taken |
| 4 | stf1.setStrLastName("patterson");  assertNotNull(stf1.getStrLastName()); | Pass | Pass | N/A | None taken |
| 5 | assertNotNull(stf1.getIntAge()); | Pass | Pass | N/A | None taken |
| 6 | stf1.setIntAge(22);  assertNotNull(stf1.getIntAge()); | Pass | Pass | N/A | None taken |
| 7 | assertNotNull(stf1.getStrTelephone()); | Pass | Pass | N/A | None taken |
| 8 | stf1.setStrTelephone("01010 124123");  assertNotNull(stf1.getStrUsername()); | Pass | Pass | N/A | None taken |
| 9 | testStaff = new Staff();  assertNotNull(testStaff); | Pass | Pass | N/A | None taken |
| 10 | testStaff = new Staff("Staff");  assertNotNull(testStaff); | Pass | Pass | N/A | None taken |
| 11 | testStaff = new Staff("Scott", "Dennison", 20, "01010 231231", "Crew Member", "sd1", "sd1"); | Pass | Pass | N/A | None taken |
| 12 | staff.addStaff(stf1); | Pass | Pass | N/A | None taken |
| 13 | staff.viewStaff(); | Pass | Pass | N/A | None taken |
| 14 | assertNotNull(stf1.getStaffID()); | Pass | Pass | N/A | None taken |
| 15 | assertNotNull(stf1.generateStaffID()); | Pass | Pass | N/A | None taken |
| 16 | assertNotNull(stf1.getStrRole()); | Pass | Pass | N/A | None taken |
| 17 | stf1.setStrRole("THE BOSS");  assertNotNull(stf1.getStrRole());  System.out.println(stf1.getStrRole()); | Pass | Pass | N/A | None taken |
| 18 | assertNotNull(stf1.getStrUsername()); | Pass | Pass | N/A | None taken |
| 19 | stf1.setStrUsername("ap\_m1");  assertNotNull(stf1.getStrUsername());  System.out.println(stf1.getStrUsername()); | Pass | Pass | N/A | None taken |
| 20 | assertNotNul(stf1.getStrPassword()); | Pass | Pass | N/A | None taken |
| 21 | stf1.setStrPassword("password");  assertNotNull(stf1.getStrPassword());  System.out.println(stf1.getStrPassword()); | Pass | Pass | N/A | None taken |
| 22 | System.out.println(stf1); | Pass | Pass | N/A | None taken |

# Unit Test – Film Date: 30.03.2011 – JUnit – Whitebox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | Film films = new Film("Films");  Film testFilm;  Film newFilm2 = new Film("Harry Potter and the Mysterious Case of Missing Work", 15); |  |  |  |  |
| 1 | assertNotNull(testFilm = new Film("Films")); | Pass | Pass | N/A | None taken |
| 2 | assertNotNull(testFilm = new Film()); | Pass | Pass | N/A | None taken |
| 3 | assertNotNull(testFilm = new Film("The Exorcist", 18)); | Pass | Pass | N/A | None taken |
| 4 | films.addFilm(newFilm2);  assertNotNull(films); | Pass | Pass | N/A | None taken |
| 5 | assertNotNull(newFilm2.getIntFilmID()); | Pass | Pass | N/A | None taken |
| 6 | newFilm2.setStrFilmName("The Exorcist 2");  assertNotNull(newFilm2.getStrFilmName());  System.out.println(newFilm2.getStrFilmName()); | Pass | Pass | N/A | None taken |
| 7 | assertNotNull(newFilm2.getStrFilmName()); | Pass | Pass | N/A | None taken |
| 8 | newFilm2.setIntRating(15);  assertNotNull(newFilm2.getIntRating());  System.out.println(newFilm2.getIntRating()); | Pass | Pass | N/A | None taken |
| 9 | assertNotNull(newFilm2.getIntRating()); | Pass | Pass | N/A | None taken |
| 10 | System.out.println(newFilm2); | Pass | Pass | N/A | None taken |
| 11 | assertNotNull(newFilm2.generateFilmID()); | Pass | Pass | N/A | None taken |

# Unit Test – Main Date: 30.03.2011 – JUnit – Whitebox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Data** | **Expected** | **Actual** | **Cause** | **Action Taken** |
| 0 | 1 – admin, admin  2 – y, 1.8  3 – user, user  4 – Harry Potter, 15  5 – Dave, Russell, 21, 01231 123132, 2  6 - 2 |  |  |  |  |
| 1 | assertNotNull(main\_package.Main.loginAdminFunction()); | Pass | Pass | N/A | None taken |
| 2 | assertNotNull(main\_package.Main.premiumSeatPrice()); | Pass | Pass | N/A | None taken |
| 3 | assertNotNull(main\_package.Main.loginUserFunction()); | Pass | Pass | N/A | None taken |
| 4 | assertNotNull(main\_package.Main.addFilmFunction()); | Pass | Pass | N/A | None taken |
| 5 | try {  assertTrue(main\_package.Main.advancedBooking());  } catch (Exception e) {  fail("Will fail regardless. Has no showings to reference");  } | Fail | Fail | N/A | None taken |
| 6 | try {  assertTrue(main\_package.Main.makeBooking());  } catch (Exception e) {  fail("Will fail regardless. Has no showings to reference");  } | Fail | Fail | N/A | Test method separately |