Atli Fannar Franklín
Jón Freyr Bjarnason
Ólafur Einar Ómarsson

Tests
- Assignment 4 in HBV 401G, team 8H -

Supervisor: Matthias Book
March 31. 2019

1 Introduction

The members of team 8H are: Atli Fannar Franklín kt. 261298-2609, Jón Freyr Bjarnason kt. 050993-2649, and Ólafur Einar Ómarsson kt. 200991-2739. Jón Freyr will be presenting the assignment in class.

2 A Test fixture

```
public class BookingTest{
    private BookingRegistry test;
    private Booking testBooking;
    private Booking testBooking1;
    private Room testRoom;
    private LocalDate date;
    @Before
    public void setUp() {
        date = LocalDate.of(Integer.parseInt("2018"), Integer.parseInt("1"), Integer.parseInt("23"));
        testBooking = new Booking(1, 1, 1, date, date, "Nothing", null, null);
        testBooking1 = new Booking(2, 1, 1, date, date, "Nothing", null, null);
        testBooking2 = new Booking(3, 1, 1, date, date, "Nothing", null, null);
        testRoom = new Room(null, 1, 0, 0, 0, 0, false, false, false, false, false, false, false, false, false, false, 0);
    }
    @After
    public void tearDown(){
        testBooking = null;
        testBooking1 = null;
        testBooking2 = null;
        testRoom = null;
    }
    @Test
    public void getBookingsTest() {
        //Perform test to see if it adds new room
        //to the bookings
         ArrayList<Booking> a = test.getBookings(date, date, testRoom);
         int oldSize = a.size();

         test.addBooking(testBooking2);

         ArrayList<Booking> b = test.getBookings(date, date, testRoom);
         assertTrue(oldSize < b.size());

    }
    @Test
    public void getBookingTest(){
        //Making a test to the booking with
        //the id 1
        Booking test1 = test.getBooking(1);
        assertTrue("True",test1 != null);

        // Making a test to see that there
        // is no Booking with id -1
        Booking test2 = test.getBooking(-1);
        assertFalse("False",test2 != null);


        String rettRequest = "Nothing";
        Booking test3 = test.getBooking(1);
        assertTrue(rettRequest.equals(test3.getRequests()));
    }
    @Test
    public void removBookingTest(){
        //Perform test to removeBooking
        boolean b = test.removeBooking(1);
        assertTrue(b);

    }
    @Test
    public void addBookingTest(){
        test.addBooking(testBooking);
        test.addBooking(testBooking1);
        Booking b = test.getBooking(1);

        // Perform actions to be tested on the booking
        assertEquals(testBooking.getId(), b.getId());
        assertEquals(testBooking.getUserId(),b.getUserId());
        assertEquals(testBooking.getRoomId(), b.getRoomId());
        assertEquals(testBooking.getFromDate(), b.getFromDate());
        assertEquals(testBooking.getToDate(), b.getToDate());
        assertEquals(testBooking.getRequests(), b.getRequests());

    }
}
```

3 Test subject

```java
public class BookingRegistry {

    public static ArrayList<Booking> getBookings(LocalDate fromDate, LocalDate toDate, Room room) {
        try {
            Class.forName("org.sqlite.JDBC");
        } catch(ClassNotFoundException e) {
            return null;
        }
        ArrayList<Booking> res = new ArrayList<Booking>();
        Connection conn = null;
        try {
            conn = DriverManager.getConnection("jdbc:sqlite:data.db");
            PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM BookingRegistry WHERE (fromDate " +
                    "<= ? OR toDate >= ?) AND roomId = ?");
            pstmt.setDate(1, java.sql.Date.valueOf(toDate));
            pstmt.setDate(2, java.sql.Date.valueOf(fromDate));
            pstmt.setInt(3, room.getId());
            ResultSet r = pstmt.executeQuery();
            while(r.next()) {
                int _id = r.getInt(1);
                int _userId = r.getInt(2);
                int _roomId = r.getInt(3);
                LocalDate _fromDate = r.getDate(4).toLocalDate();
                LocalDate _toDate = r.getDate(5).toLocalDate();
                String _requests = r.getString(6);
                User booker = UserRegistry.getUser(_userId);
                Room booked = RoomRegistry.getRoom(_roomId);
                Booking result = new Booking(_id, _userId, _roomId, _fromDate, _toDate, _requests, booker, booked);
                res.add(result);
            }
        } catch(SQLException e) {
            res = null;
        } finally {
            try {
                if(conn != null) {
                    conn.close();
                }
            } catch(SQLException e) {
                res = null;
            } finally {
                return res;
            }
        }
    }
```

```java
public static Booking getBooking(int id) {
    try {
        Class.forName("org.sqlite.JDBC");
    } catch(ClassNotFoundException e) {
        return null;
    }
    Connection conn = null;
    try {
        conn = DriverManager.getConnection("jdbc:sqlite:data.db");
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM BookingRegistry WHERE id = ?");
        pstmt.setInt(1, id);
        ResultSet r = pstmt.executeQuery();
        if(!r.next()) {
            return null;
        }
        int _id = r.getInt(1);
        int _userId = r.getInt(2);
        int _roomId = r.getInt(3);
        LocalDate _fromDate = r.getDate(4).toLocalDate();
        LocalDate _toDate = r.getDate(5).toLocalDate();
        String _requests = r.getString(6);
        User booker = UserRegistry.getUser(_userId);
        Room booked = RoomRegistry.getRoom(_roomId);
        Booking result = new Booking(_id, _userId, _roomId, _fromDate, _toDate, _requests, booker, booked);
        try {
            if(conn != null) {
                conn.close();
            }
        } catch(SQLException e) {
            return null;
        }
        return result;
    } catch(SQLException e) {
        return null;
    }
}
```

```java
public static boolean removeBooking(int id) {
    try {
        Class.forName("org.sqlite.JDBC");
    } catch(ClassNotFoundException e) {
        return false;
    }
    Connection conn = null;
    try {
        conn = DriverManager.getConnection("jdbc:sqlite:data.db");
        PreparedStatement pstmt = conn.prepareStatement("DELETE FROM BookingRegistry WHERE id = ?");
        pstmt.setInt(1, id);
        pstmt.executeUpdate();
        try {
            if(conn != null) {
                conn.close();
            }
        } catch(SQLException e) {
            return false;
        }
        return true;
    } catch(SQLException e) {
        return false;
    }
}

public static boolean addBooking(Booking book) {
    try {
        Class.forName("org.sqlite.JDBC");
    } catch(ClassNotFoundException e) {
        return false;
    }
    Connection conn = null;
    try {
        conn = DriverManager.getConnection("jdbc:sqlite:data.db");
        PreparedStatement pstmt = conn.prepareStatement("INSERT INTO BookingRegistry VALUES (?,?,?,?,?,?)");
        pstmt.setInt(1, book.getId());
        pstmt.setInt(2, book.getUserId());
        pstmt.setInt(3, book.getRoomId());
        pstmt.setDate(4, java.sql.Date.valueOf(book.getFromDate()));
        pstmt.setDate(5, java.sql.Date.valueOf(book.getToDate()));
        pstmt.setString(6, book.getRequests());
        pstmt.executeUpdate();
        try {
            if(conn != null) {
                conn.close();
            }
        } catch(SQLException e) {
            return false;
        }
        return true;
    } catch(SQLException e) {
        return false;
    }
}
```