Continuing Education Program
Centre for Computing and Information Technology
Faculty of Engineering – University of Indonesia
Tel: 021-7863508 Mail: ccit.@eng.ui.ac.id

TRAVELNOW TESTING

Created By:

Christian Frans Mukuan (1920010040)
Deyaninta Ekabriela Permata (1920010050)

Class
3SC3

Faculty:
Riza Muhammad Nurman S.Kom

CEP CCIT

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

2020

PROJECT ON

TravelNow Testing

Developed by :

1. Christian Frans Mukuan (1920010040)
2. Deyaninta Ekabriela Permata (1920010050)

.

TravelNow Testing

Batch Code	: 3SC3 (Group 4)
Start Date	: October, 2020
End Date	: November 18, 2020
Date of Submission	: November 19, 2020
Name of Faculty	**:** Riza Muhammad Nurman S.Kom
Names of Developer	:

1. Christian Frans Mukuan (1920010040)
2. Deyaninta Ekabriela Permata (1920010150)

.

NIIT

CERTIFICATE

This is to certify that this report titled “TravelNow Testing” embodies the original work done by Christian Frans Mukuan, and Deyaninta Ekabriela Permata in partial fulfillment of their course requirement at NIIT.

Coordinator :

Riza Muhammad Nurman S.Kom

.

ACKNOWLEDGEMENT

First of all, thanks to Allah SWT because we can complete this Project task both in the form of a presentation. We want to deliver sincere especially for Mr. Riza Muhamad Nurman’s faculty and another faculty who always help. Thank you also to fellow students who have supported, and also thank you for being fellow workers in the education at CCIT-FTUI.

The Project paper entitled ‘TravelNow Testing' the writers submits as Project's 2020 task requirements.

The hope of the writers hopefully this paper can be useful for all so that it can add knowledge and insight. The writers realize that this paper is far from perfect. Therefore, the writers expect all suggestions and criticisms from readers who are constructive for the perfection of this paper.

Depok, November 2020

Writers

.

SYSTEM ANALYSIS

System Summary :

TravelNow is an Indonesian startup company that provides flight ticket booking services, hotel tickets, tour tickets, transportation services that are complemented by tour guide services to accompany customers on international holidays to several Asian countries. Based on operations in Jakarta, TravelNow is focused on helping to manage the needs of Indonesian public holidays or business trips.

.

CONFIGURATION

. **Hardware** :

ASUS A442UR Intel® Core™ i5-8250U

Operating System :

Windows 10 Home Single Language 64-bit (10.0, Build 18362)

Software :

Netbeans IDE 8.2

.

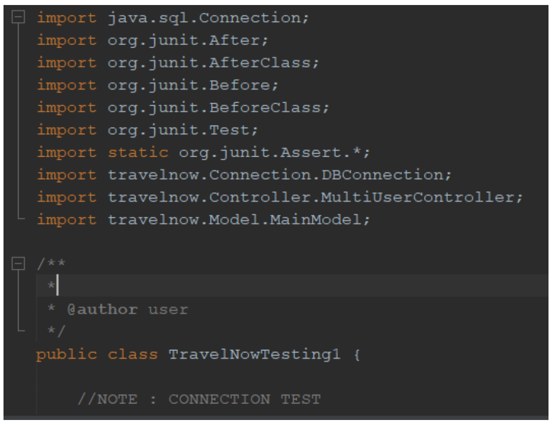
TYPE OF TESTING TOOLS

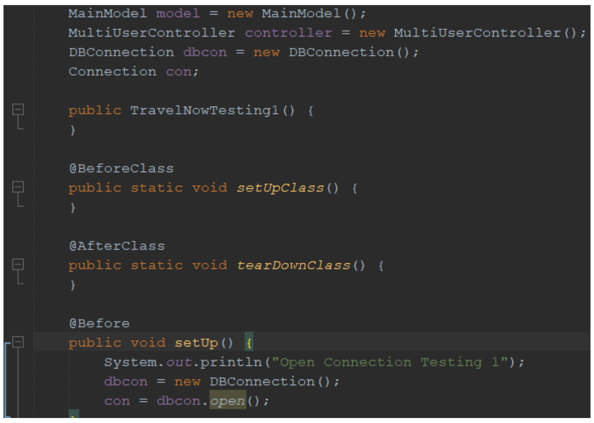
- JUnit JUnit is a unit testing framework for java. Used and writed by developer for testing a specific area of a functionality of the code to be tested.
- JaCoCo JaCoCo is a free code coverage library for Java, which has been created by the EclEmma team based on the lessons learned from using and integration existing libraries for many years.
- Black Box Testing Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester

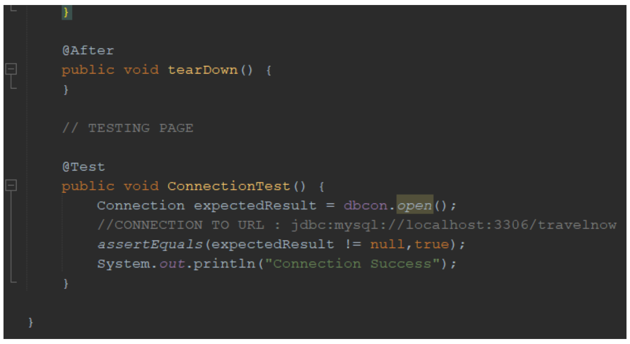
.

TESTING IMPLEMENTATION

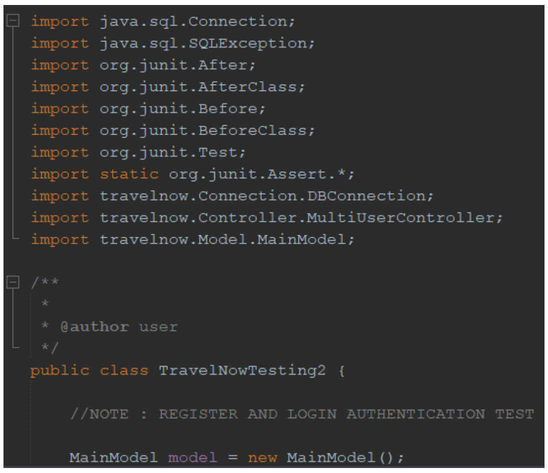
TravelNow Testing 1

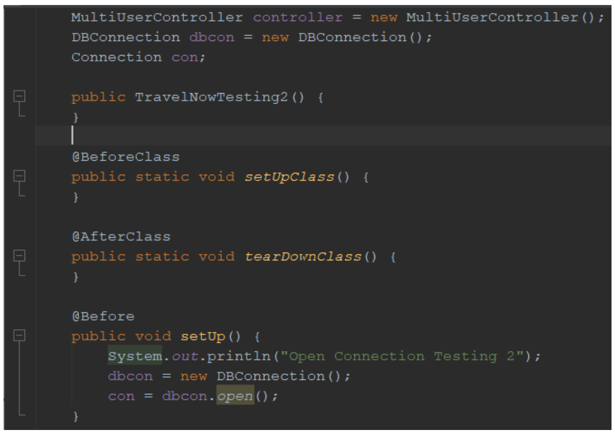

```
import java.sql.Connection;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
import travelnow.Connection.DBConnection;
import travelnow.Controller.MultiUserController;
import travelnow.Model.MainModel;

/**
 *
 * @author user
 */
public class TravelNowTesting1 {

    //NOTE : CONNECTION TEST
```


```
    MainModel model = new MainModel();
    MultiUserController controller = new MultiUserController();
    DBConnection dbcon = new DBConnection();
    Connection con;

    public TravelNowTesting1() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
        System.out.println("Open Connection Testing 1");
        dbcon = new DBConnection();
        con = dbcon.open();
```


```
    }

    @After
    public void tearDown() {
    }

    // TESTING PAGE

    @Test
    public void ConnectionTest() {
        Connection expectedResult = dbcon.open();
        //CONNECTION TO URL : jdbc:mysql://localhost:3306/travelnow
        assertEquals(expectedResult != null,true);
        System.out.println("Connection Success");
    }

}
```

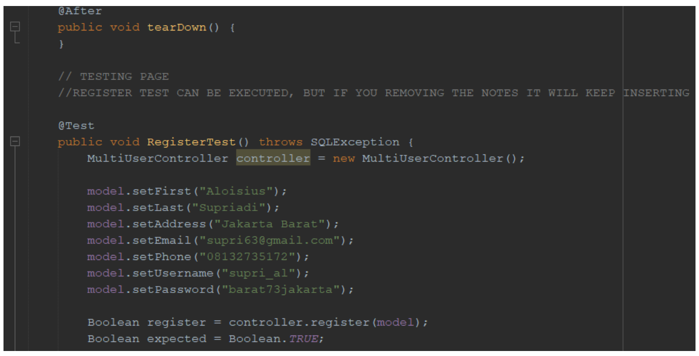
TravelNow Testing 2


```
import java.sql.Connection;
import java.sql.SQLException;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
import travelnow.Connection.DBConnection;
import travelnow.Controller.MultiUserController;
import travelnow.Model.MainModel;

/**
 *
 * @author user
 */
public class TravelNowTesting2 {

    //NOTE : REGISTER AND LOGIN AUTHENTICATION TEST

    MainModel model = new MainModel();
```

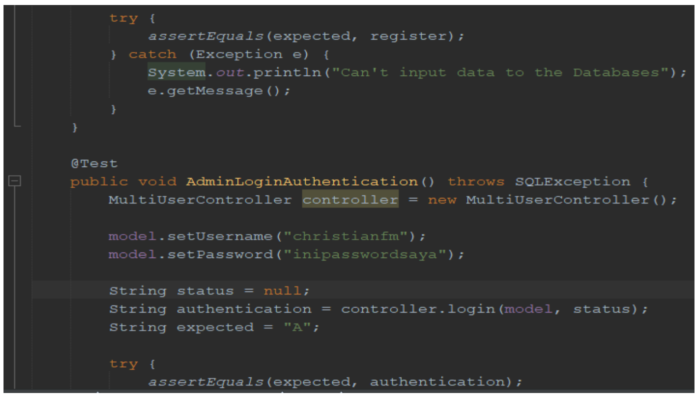

```
    MultiUserController controller = new MultiUserController();
    DBConnection dbcon = new DBConnection();
    Connection con;

    public TravelNowTesting2() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
        System.out.println("Open Connection Testing 2");
        dbcon = new DBConnection();
        con = dbcon.open();
    }
```

.

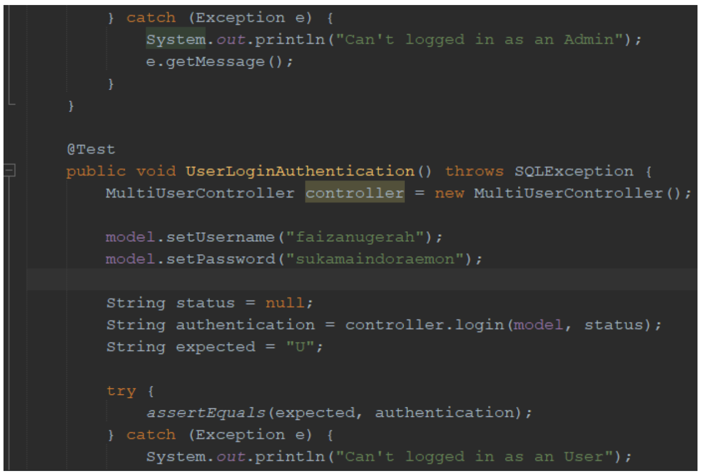
TESTING IMPLEMENTATION



```
@After
public void tearDown() {
}

// TESTING PAGE
//REGISTER TEST CAN BE EXECUTED, BUT IF YOU REMOVING THE NOTES IT WILL KEEP INSERTING

@Test
public void RegisterTest() throws SQLException {
    MultiUserController controller = new MultiUserController();

    model.setFirst("Aloisius");
    model.setLast("Supriadi");
    model.setAddress("Jakarta Barat");
    model.setEmail("supri63@gmail.com");
    model.setPhone("08132735172");
    model.setUsername("supri_al");
    model.setPassword("barat73jakarta");

    Boolean register = controller.register(model);
    Boolean expected = Boolean.TRUE;
```



```
        try {
            assertEquals(expected, register);
        } catch (Exception e) {
            System.out.println("Can't input data to the Databases");
            e.getMessage();
        }
    }

    @Test
    public void AdminLoginAuthentication() throws SQLException {
        MultiUserController controller = new MultiUserController();

        model.setUsername("christianfm");
        model.setPassword("inipasswordsaya");

        String status = null;
        String authentication = controller.login(model, status);
        String expected = "A";

        try {
            assertEquals(expected, authentication);
```

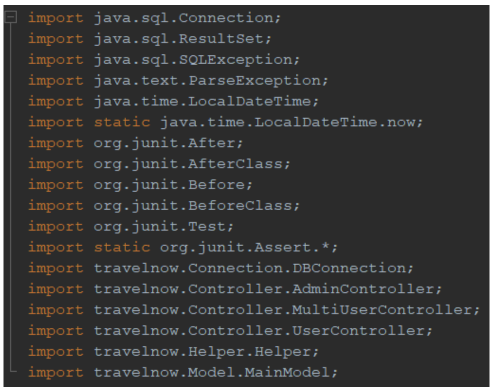


```
        } catch (Exception e) {
            System.out.println("Can't logged in as an Admin");
            e.getMessage();
        }
    }

    @Test
    public void UserLoginAuthentication() throws SQLException {
        MultiUserController controller = new MultiUserController();

        model.setUsername("faizanugerah");
        model.setPassword("sukamaindoraemon");

        String status = null;
        String authentication = controller.login(model, status);
        String expected = "U";

        try {
            assertEquals(expected, authentication);
        } catch (Exception e) {
            System.out.println("Can't logged in as an User");
```

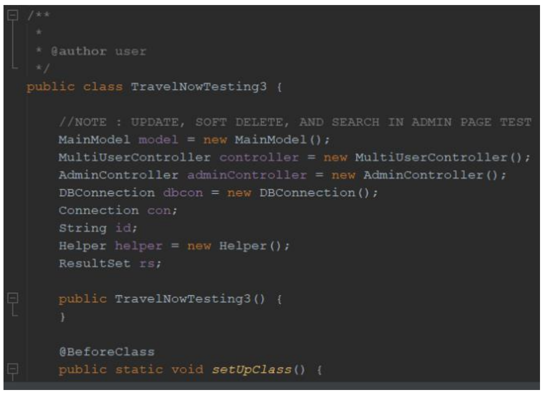
.

TESTING IMPLEMENTATION

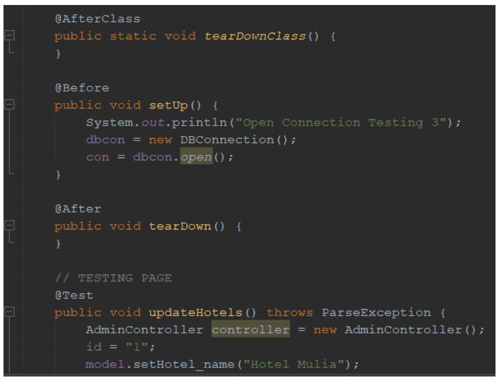
TravelNow Testing 3



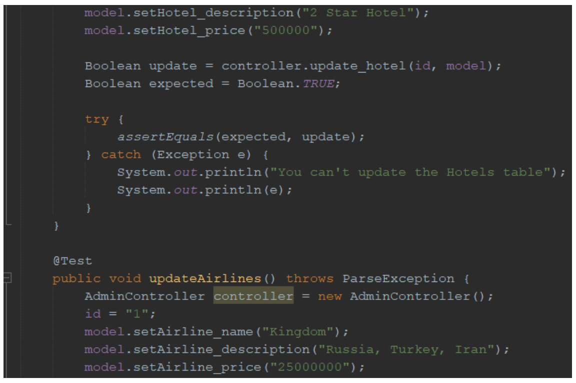
```java
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.time.LocalDateTime;
import static java.time.LocalDateTime.now;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
import travelnow.Connection.DBConnection;
import travelnow.Controller.AdminController;
import travelnow.Controller.MultiUserController;
import travelnow.Controller.UserController;
import travelnow.Helper.Helper;
import travelnow.Model.MainModel;
```



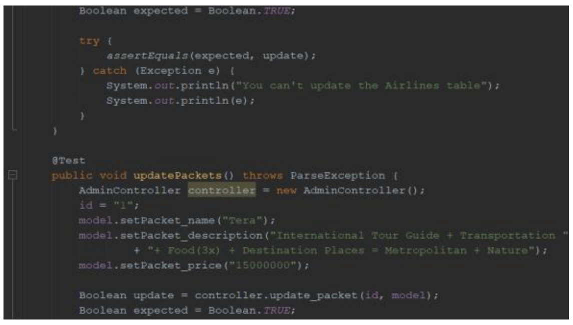
```java
/**
 *
 * @author user
 */
public class TravelNowTesting3 {

    //NOTE : UPDATE, SOFT DELETE, AND SEARCH IN ADMIN PAGE TEST
    MainModel model = new MainModel();
    MultiUserController controller = new MultiUserController();
    AdminController adminController = new AdminController();
    DBConnection dbcon = new DBConnection();
    Connection con;
    String id;
    Helper helper = new Helper();
    ResultSet rs;

    public TravelNowTesting3() {
    }

    @BeforeClass
    public static void setUpClass() {
```



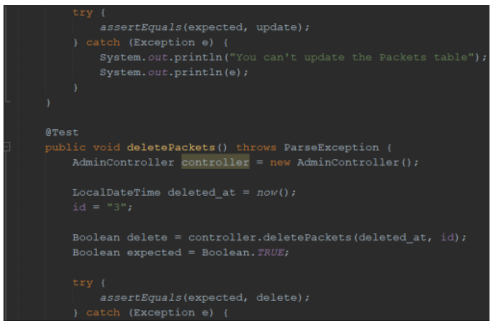
```java
    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
        System.out.println("Open Connection Testing 3");
        dbcon = new DBConnection();
        con = dbcon.open();
    }

    @After
    public void tearDown() {
    }

    // TESTING PAGE
    @Test
    public void updateHotels() throws ParseException {
        AdminController controller = new AdminController();
        id = "1";
        model.setHotel_name("Hotel Mulia");
```



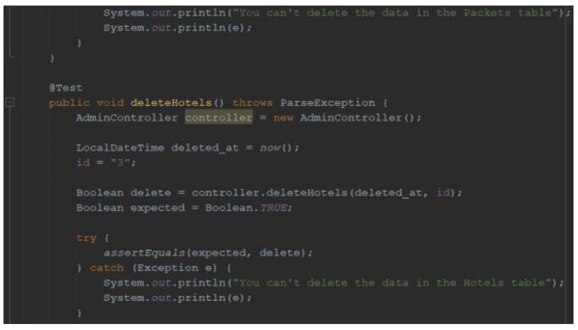
```java
        model.setHotel_description("2 Star Hotel");
        model.setHotel_price("500000");

        Boolean update = controller.update_hotel(id, model);
        Boolean expected = Boolean.TRUE;

        try {
            assertEquals(expected, update);
        } catch (Exception e) {
            System.out.println("You can't update the Hotels table");
            System.out.println(e);
        }
    }

    @Test
    public void updateAirlines() throws ParseException {
        AdminController controller = new AdminController();
        id = "1";
        model.setAirline_name("Kingdom");
        model.setAirline_description("Russia, Turkey, Iran");
        model.setAirline_price("25000000");
```



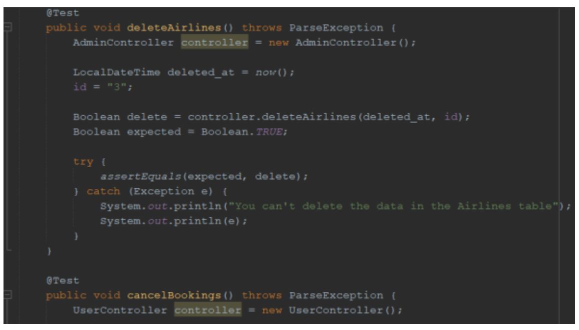
```java
        Boolean expected = Boolean.TRUE;

        try {
            assertEquals(expected, update);
        } catch (Exception e) {
            System.out.println("You can't update the Airlines table");
            System.out.println(e);
        }
    }

    @Test
    public void updatePackets() throws ParseException {
        AdminController controller = new AdminController();
        id = "1";
        model.setPacket_name("Tera");
        model.setPacket_description("International Tour Guide + Transportation "
                + "+ Food(3x) + Destination Places = Metropolitan + Nature");
        model.setPacket_price("15000000");

        Boolean update = controller.update_packet(id, model);
        Boolean expected = Boolean.TRUE;
```



```java
        try {
            assertEquals(expected, update);
        } catch (Exception e) {
            System.out.println("You can't update the Packets table");
            System.out.println(e);
        }
    }

    @Test
    public void deletePackets() throws ParseException {
        AdminController controller = new AdminController();

        LocalDateTime deleted_at = now();
        id = "3";

        Boolean delete = controller.deletePackets(deleted_at, id);
        Boolean expected = Boolean.TRUE;

        try {
            assertEquals(expected, delete);
        } catch (Exception e) {
```



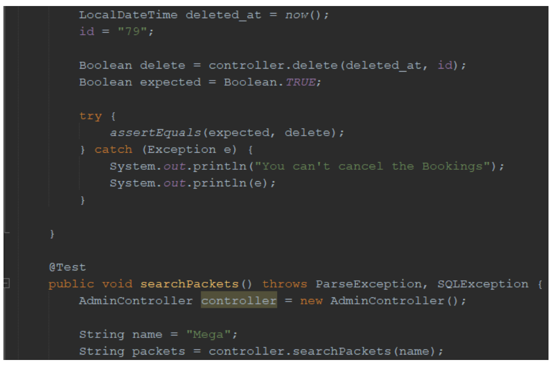
```java
            System.out.println("You can't delete the data in the Packets table");
            System.out.println(e);
        }
    }

    @Test
    public void deleteHotels() throws ParseException {
        AdminController controller = new AdminController();

        LocalDateTime deleted_at = now();
        id = "3";

        Boolean delete = controller.deleteHotels(deleted_at, id);
        Boolean expected = Boolean.TRUE;

        try {
            assertEquals(expected, delete);
        } catch (Exception e) {
            System.out.println("You can't delete the data in the Hotels table");
            System.out.println(e);
        }
```



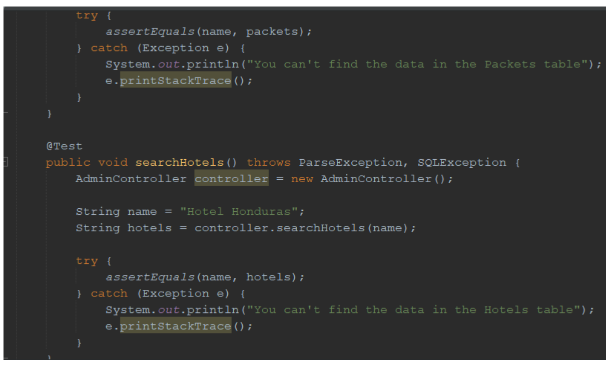
```java
    @Test
    public void deleteAirlines() throws ParseException {
        AdminController controller = new AdminController();

        LocalDateTime deleted_at = now();
        id = "3";

        Boolean delete = controller.deleteAirlines(deleted_at, id);
        Boolean expected = Boolean.TRUE;

        try {
            assertEquals(expected, delete);
        } catch (Exception e) {
            System.out.println("You can't delete the data in the Airlines table");
            System.out.println(e);
        }
    }

    @Test
    public void cancelBookings() throws ParseException {
        UserController controller = new UserController();
```

.

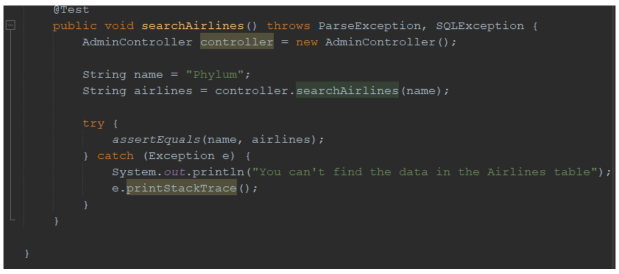
TESTING IMPLEMENTATION



```java
        LocalDateTime deleted_at = now();
        id = "79";

        Boolean delete = controller.delete(deleted_at, id);
        Boolean expected = Boolean.TRUE;

        try {
            assertEquals(expected, delete);
        } catch (Exception e) {
            System.out.println("You can't cancel the Bookings");
            System.out.println(e);
        }

    }

    @Test
    public void searchPackets() throws ParseException, SQLException {
        AdminController controller = new AdminController();

        String name = "Mega";
        String packets = controller.searchPackets(name);
```

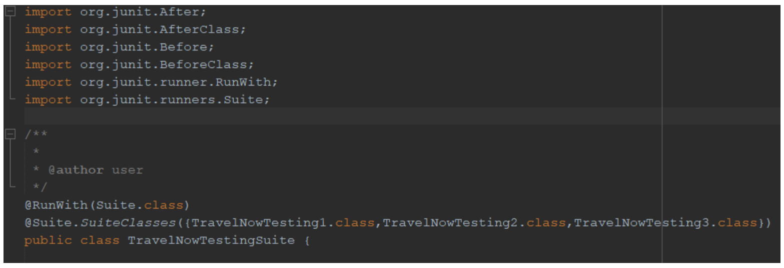


```java
        try {
            assertEquals(name, packets);
        } catch (Exception e) {
            System.out.println("You can't find the data in the Packets table");
            e.printStackTrace();
        }
    }

    @Test
    public void searchHotels() throws ParseException, SQLException {
        AdminController controller = new AdminController();

        String name = "Hotel Honduras";
        String hotels = controller.searchHotels(name);

        try {
            assertEquals(name, hotels);
        } catch (Exception e) {
            System.out.println("You can't find the data in the Hotels table");
            e.printStackTrace();
        }
```

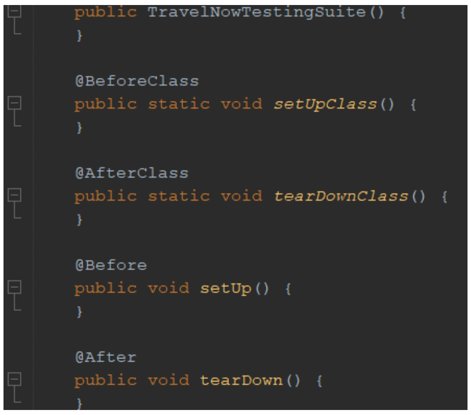


```java
    @Test
    public void searchAirlines() throws ParseException, SQLException {
        AdminController controller = new AdminController();

        String name = "Phylum";
        String airlines = controller.searchAirlines(name);

        try {
            assertEquals(name, airlines);
        } catch (Exception e) {
            System.out.println("You can't find the data in the Airlines table");
            e.printStackTrace();
        }
    }

}
```

TravelNow Tesing Suite



```java
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

/**
 *
 * @author user
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({TravelNowTesting1.class,TravelNowTesting2.class,TravelNowTesting3.class})
public class TravelNowTestingSuite {
```



```java
    public TravelNowTestingSuite() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }
```

.

TESTING IMPLEMENTATION

No.	Test Case	Test Data	Expected Result	Actual Result	Result
1.	Verification Login for Users	Using valid id and password	Go to the users page	Go to the users page and show “Login Success”	PASS
2.	Verification Login for Users	Using invalid id and password	Go to the login page	Go to the login page and show “Login Failed”	PASS
3.	Verification Login for Admin	Using valid Id and password	Go to the admin panel page	Go to the admin panel page	PASS
4.	Verification Login for Admin	Using invalid id and password	Go to the login page	Go to the login page and show “Login Failed”	PASS
5.	Registration account for users	Fill the users data in the register page	Your data have been submitted	Your data have been submitted and go the login page	PASS
6.	Submit Button	To make a bookings in the users page	Your bookings success	Your data have been submitted and go the passengers details page	PASS
7.	Submit button	To fill the data passengers details	Your data have been submitted	Your data have been submitted	PASS

.

TESTING IMPLEMENTATION

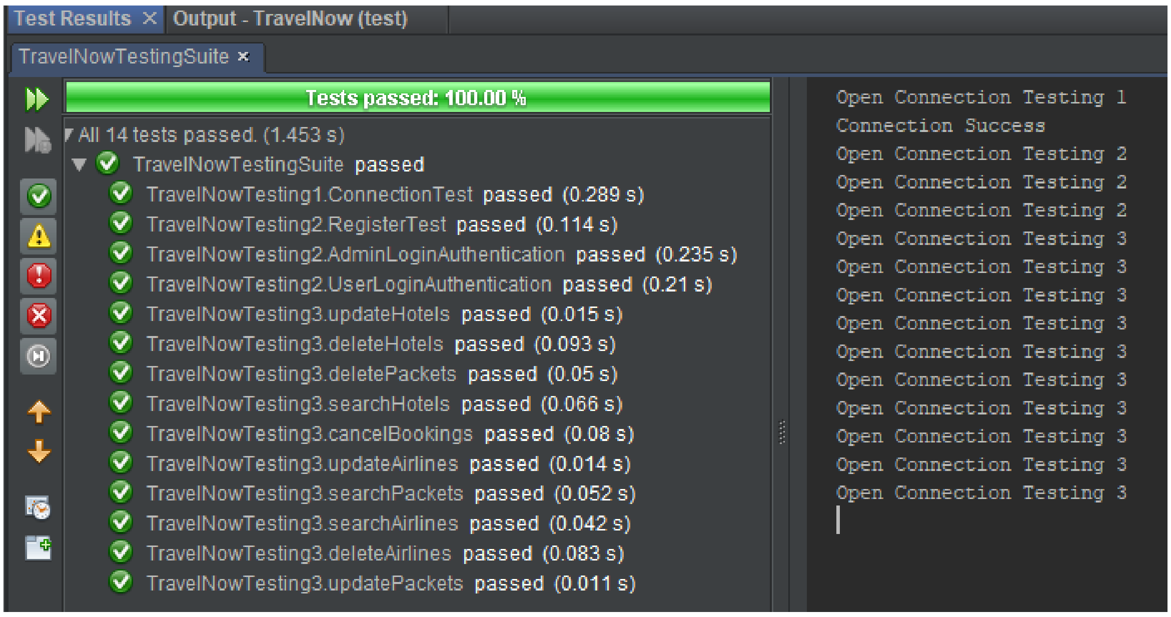
No.	Test Case	Test Data	Expected Result	Actual Result	Result
8.	Delete the user’s bookings data	Delete data bookings by clicking the cancel button	The data bookings have been deleted	The data bookings have been deleted	PASS
9.	Delete the hotel data on the table hotel list	Admin can choose one of the hotel data and click the delete button	The hotel data have been deleted	The hotel data have been deleted	PASS
10.	Delete the airlines data on the table airlines list	Admin can choose one of the airlines data and click the delete button	The airlines data have been deleted	The airlines data have been deleted	PASS
11.	Delete the packets data on the table packets list	Admin can choose one of the packets data and click the delete button	The packets data have been deleted	The packets data have been deleted	PASS
12.	Update the hotel data on the table hotel list	Admin can update the hotel data on the text area and click the update button	The hotels data have been updated	The hotels data have been updated	PASS
13.	Update the airlines data on the table airlines list	Admin can update the airlines data on the text area and click the update button	The airlines data have been updated	The airlines data have been updated	PASS

TESTING IMPLEMENTATION

No.	Test Case	Test Data	Expected Result	Actual Result	Result
14.	Update the packets data on the table airlines list	Admin can update the packets data on the text area and click the update button	The packets data have been updated	The packets data have been updated	PASS
15.	Create the new hotel data on the table hotel list	Admin can add the new hotel data on the text area	The hotel data have been added	The hotel data have been added	PASS
16.	Create the new airlines data on the table airlines list	Admin can add the new airlines data on the text area	The airlines data have been added	The airlines data have been added	PASS
17.	Create the new packets data on the table packets list	Admin can add the new packets data on the text area	The packets data have been deleted	The packets data have been deleted	PASS
18.	Search By Name button	Admin can search the hotel, airlines, and packets data by the name	The data has appeared	The data has appeared	PASS
19.	Search by Price Button	Admin can search the hotel, airlines, and packets data by the price	The data has appeared	The data has appeared	PASS
20.	Log out button	log out the account by clicking the log out button	Show the message “Are you sure?” and click ok butotn	Show the message “Are you sure?” and click ok butotn	PASS

S

RESULT



Test Results × Output - TravelNow (test)

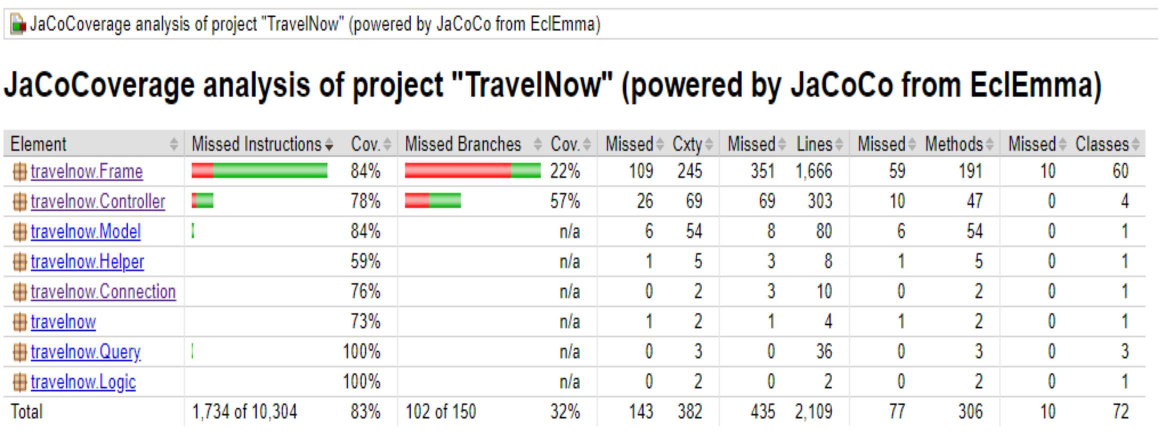
TravelNowTestingSuite ×

Tests passed: 100.00 %

```
All 14 tests passed. (1.453 s)
  TravelNowTestingSuite passed
    TravelNowTesting1.ConnectionTest passed (0.289 s)
    TravelNowTesting2.RegisterTest passed (0.114 s)
    TravelNowTesting2.AdminLoginAuthentication passed (0.235 s)
    TravelNowTesting2.UserLoginAuthentication passed (0.21 s)
    TravelNowTesting3.updateHotels passed (0.015 s)
    TravelNowTesting3.deleteHotels passed (0.093 s)
    TravelNowTesting3.deletePackets passed (0.05 s)
    TravelNowTesting3.searchHotels passed (0.066 s)
    TravelNowTesting3.cancelBookings passed (0.08 s)
    TravelNowTesting3.updateAirlines passed (0.014 s)
    TravelNowTesting3.searchPackets passed (0.052 s)
    TravelNowTesting3.searchAirlines passed (0.042 s)
    TravelNowTesting3.deleteAirlines passed (0.083 s)
    TravelNowTesting3.updatePackets passed (0.011 s)
```

```
Open Connection Testing 1
Connection Success
Open Connection Testing 2
Open Connection Testing 2
Open Connection Testing 2
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
Open Connection Testing 3
```

RESULT JaCoCo



JaCoCoverage analysis of project "TravelNow" (powered by JaCoCo from EclEmma)

JaCoCoverage analysis of project "TravelNow" (powered by JaCoCo from EclEmma)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
travelnow.Frame		84%		22%	109	245	351	1,666	59	191	10	60
travelnow.Controller		78%		57%	26	69	69	303	10	47	0	4
travelnow.Model		84%		n/a	6	54	8	80	6	54	0	1
travelnow.Helper		59%		n/a	1	5	3	8	1	5	0	1
travelnow.Connection		76%		n/a	0	2	3	10	0	2	0	1
travelnow		73%		n/a	1	2	1	4	1	2	0	1
travelnow.Query		100%		n/a	0	3	0	36	0	3	0	3
travelnow.Logic		100%		n/a	0	2	0	2	0	2	0	1
Total	1,734 of 10,304	83%	102 of 150	32%	143	382	435	2,109	77	306	10	72