

```
1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.exception.JDBCDataBaseException;
4 import il.ac.hit.CashFlowManagement.model.JDBCDataBase;
5 import il.ac.hit.CashFlowManagement.model.UserLogic;
6 import org.junit.After;
7 import org.junit.Before;
8 import org.junit.Test;
9
10 import java.sql.SQLException;
11
12 import static org.junit.Assert.assertEquals;
13 import static org.junit.Assert.assertTrue;
14
15 public class UserLogicTest {
16     UserLogic userLogic = new UserLogic();
17     String userName = "testUserName", password = "testPassword",
18     country = "testCountry", gander = "testGander";
19
20     @Before
21     public void setUp() {
22         userLogic.addUser(userName, password, country, gander);
23     }
24
25     @After
26     public void tearDown() throws JDBCDataBaseException, SQLException
27     {
28         while(userLogic.checkIfExist("userName") || userLogic.
29         checkIfExist(userName)) {
30             JDBCDataBase.getInstance().getStatement().executeUpdate("
31             DELETE FROM Registered_Users WHERE UserName = 'testUserName'");
32             JDBCDataBase.getInstance().getStatement().executeUpdate("
33             DELETE FROM Registered_Users WHERE UserName = 'userName'");
34         }
35     }
36
37     @Test
38     public void checkIfUserExist() throws JDBCDataBaseException,
39     SQLException {
40         assertEquals(false, userLogic.checkIfExist("userName"));
41         assertTrue(userLogic.checkIfExist(userName));
42     }
43
44     @Test
45     public void addUser() {
```

```
41         assertEquals(false, userLogic.checkIfExist("userName"));
42         userLogic.addUser("userName", password, country, gander);
43         assertTrue(userLogic.checkIfExist("userName"));
44     }
45
46     @Test
47     public void checkUserPassword() {
48         assertEquals(true, userLogic.checkUserPassword(userName,
49 password));
50         assertEquals(false, userLogic.checkUserPassword(userName,
51 password + "1"));
52     }
53 }
```

```
1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.exception.JDBCDataBaseException;
4 import il.ac.hit.CashFlowManagement.model.JDBCDataBase;
5 import org.apache.log4j.BasicConfigurator;
6 import org.junit.Test;
7
8 import static junit.framework.TestCase.assertNotNull;
9
10 public class JDBCDataBaseTest {
11     public JDBCDataBaseTest(){
12         BasicConfigurator.configure();
13     }
14
15     @Test
16     public void getInstance() {
17         try {
18             assertNotNull(JDBCDataBase.getInstance());
19         } catch (JDBCDataBaseException e) {
20             e.printStackTrace();
21         }
22     }
23
24     @Test
25     public void getStatement() {
26         try {
27             assertNotNull(JDBCDataBase.getInstance().getStatement());
28         } catch (JDBCDataBaseException e) {
29             e.printStackTrace();
30         }
31     }
32
33     @Test
34     public void getConnection() {
35         try {
36             assertNotNull(JDBCDataBase.getInstance().getConnection
37 ());
38         } catch (JDBCDataBaseException e) {
39             e.printStackTrace();
40         }
41     }
42 }
43
```

```

1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.model.Expense;
4 import il.ac.hit.CashFlowManagement.model.ExpensesLogic;
5 import il.ac.hit.CashFlowManagement.model.JDBCDataBaseLogic;
6 import il.ac.hit.CashFlowManagement.view.LoginForm;
7 import il.ac.hit.CashFlowManagement.viewmodel.IViewModel;
8 import il.ac.hit.CashFlowManagement.viewmodel.ManagementViewModel;
9 import org.apache.log4j.BasicConfigurator;
10 import org.junit.After;
11 import org.junit.Test;
12
13 import java.sql.ResultSet;
14
15 import static org.junit.Assert.assertEquals;
16
17 public class ExpensesLogicTest {
18     private JDBCDataBaseLogic dataBaseLogic = JDBCDataBaseLogic.
19         getInstance();
20     private String expenseDate = "01/01/2020", expenseClassification
21         = "testClassification", expenseDetails = "testDetails";
22     private double expenseCost = 5.55;
23     private IViewModel viewModel;
24     private String nameOfTable;
25     private ExpensesLogic expensesLogic;
26     private Expense expense = new Expense(expenseDate,
27         expenseClassification, expenseDetails, expenseCost);
28
29     public ExpensesLogicTest(){
30         BasicConfigurator.configure();
31         LoginForm.username = "testUserName";
32         viewModel = ManagementViewModel.getInstance();
33         nameOfTable = LoginForm.username.toUpperCase() + "_Expenses".
34         toUpperCase();
35         expensesLogic = new ExpensesLogic();
36     }
37
38     @After
39     public void tearDown() throws Exception {
40         dataBaseLogic.removeTable(nameOfTable);
41     }
42
43     @Test
44     public void addExpense() throws Exception {
45         expensesLogic.addExpense(expense);
46         ResultSet rs = expensesLogic.getAllUserExpenses();

```

```
43         rs.next();
44         String date = rs.getString("Date");
45         String cost = rs.getString("Cost");
46         String classification = rs.getString("Classification");
47         String details = rs.getString("Details");
48         double costDouble = Double.parseDouble(cost);
49
50         assertEquals(expenseDate, date);
51         assertEquals((long)expenseCost, (long)costDouble);
52         assertEquals(expenseClassification, classification);
53         assertEquals(expenseDetails, details);
54     }
55
56     @Test
57     public void getAllUserExpenses() throws Exception {
58         int expensesCounter = 0;
59         expensesLogic.addExpense(expense);
60         ResultSet rs = expensesLogic.getAllUserExpenses();
61         while (rs.next()){
62             expensesCounter++;
63         }
64         assertEquals(1, expensesCounter);
65     }
66 }
67
```

```
1 package il.ac.hit.CashFlowManagement.test;
2
3
4 import il.ac.hit.CashFlowManagement.exception.
    JDBCDataBaseLogicException;
5 import il.ac.hit.CashFlowManagement.model.JDBCDataBaseLogic;
6 import org.apache.log4j.BasicConfigurator;
7
8 import static org.hamcrest.CoreMatchers.is;
9 import static org.junit.Assert.*;
10
11
12 public class JDBCDataBaseLogicTest {
13     private JDBCDataBaseLogic dataBaseLogic = JDBCDataBaseLogic.
    getInstance();
14     private final String iNameOfTable = "Test";
15
16     public JDBCDataBaseLogicTest(){
17         BasicConfigurator.configure();
18     }
19
20     @org.junit.Before
21     public void setUp() throws Exception {
22         dataBaseLogic.createTableIfNotExist(iNameOfTable,"Password",
    "varchar(300)");
23     }
24
25
26     @org.junit.After
27     public void tearDown() throws Exception {
28         dataBaseLogic.removeTable(iNameOfTable);
29     }
30
31     @org.junit.Test
32     public void createTableIfNotExist() throws Exception {
33         String tableName = "CreateTableTest";
34
35         assertEquals(true,dataBaseLogic.createTableIfNotExist(
    tableName,"Password", "varchar(300)"));
36         assertEquals(false,dataBaseLogic.createTableIfNotExist(
    tableName,"Password", "varchar(300)"));
37         dataBaseLogic.removeTable(tableName);
38         try{
39             dataBaseLogic.createTableIfNotExist(iNameOfTable,"
    Password");
40             fail("Expected exception has not been thrown");
```

```

41     } catch (Exception e) {
42         assertThat(e.getMessage(), is("Please enter the
parameters as follows: Param1Name ,Param1Type, Param2Name ,Param2Type
.... , ParamName ,ParamType"));
43     }
44 }
45
46 @org.junit.Test
47 public void insertValue() throws Exception {
48     String params = "" + "134" + "";
49
50     dataBaseLogic.insertValue(iNameOfTable,params);
51     try {
52         dataBaseLogic.insertValue(iNameOfTable, null);
53         // If the exception is not thrown, the test will fail
54         fail("Expected exception has not been thrown");
55     } catch (Exception e) {
56         assertThat(e.getMessage(), is("Argument for @NotNull
parameter 'iParameters' of il/ac/hit/CashFlowManagement/model/
JDBCDataBaseLogic.insertValue must not be null"));
57     }
58 }
59
60 @org.junit.Test
61 public void query() throws JDBCDataBaseLogicException {
62     assertNotNull(dataBaseLogic.query("select * from " +
iNameOfTable));
63
64     try {
65         assertNotNull(dataBaseLogic.query("select * from " +
iNameOfTable + "1"));
66         fail("Expected exception has not been thrown");
67     } catch (Exception e) {
68         assertThat(e.getMessage(), is("query execution failed
Table/View 'TEST1' does not exist."));
69     }
70
71     try {
72         dataBaseLogic.query("create table TEST (Password varchar(
300))");
73         fail("Expected exception has not been thrown");
74     } catch (Exception e) {
75         assertThat(e.getMessage(), is("query execution failed
Table/View 'TEST' already exists in Schema 'APP'."));
76     }
77 }

```

```
78
79     @org.junit.Test
80     public void removeTable() throws JDBCDataBaseLogicException {
81         String tableName = "RemoveTestTable";
82
83         dataBaseLogic.createTableIfNotExist(tableName, "Password", "
varchar(300)");
84         dataBaseLogic.removeTable(tableName);
85
86         try {
87             dataBaseLogic.removeTable(tableName+"1");
88             fail("Expected exception has not been thrown");
89         } catch (Exception e) {
90             assertThat(e.getMessage(), is("failed to remove
RemoveTestTable1 from DB"));
91         }
92     }
93
94     @org.junit.Test
95     public void getInstance() throws JDBCDataBaseLogicException {
96         assertNotNull(JDBCDataBaseLogic.getInstance());
97     }
98
99 }
```



```

1 package il.ac.hit.CashFlowManagement.test;
2
3 import il.ac.hit.CashFlowManagement.exception.FormCastingException;
4 import il.ac.hit.CashFlowManagement.exception.
    GetAllUserExpensesException;
5 import il.ac.hit.CashFlowManagement.exception.JDBCDataBaseException;
6 import il.ac.hit.CashFlowManagement.model.Expense;
7 import il.ac.hit.CashFlowManagement.model.ExpensesLogic;
8 import il.ac.hit.CashFlowManagement.model.JDBCDataBase;
9 import il.ac.hit.CashFlowManagement.model.UserLogic;
10 import il.ac.hit.CashFlowManagement.view.LoginForm;
11 import il.ac.hit.CashFlowManagement.viewmodel.ManagementViewModel;
12 import org.junit.After;
13 import org.junit.Before;
14 import org.junit.Test;
15
16 import java.sql.ResultSet;
17 import java.sql.SQLException;
18
19 import static org.junit.Assert.assertEquals;
20 import static org.junit.Assert.assertNotNull;
21
22 public class ManagementViewModelTest {
23     private ManagementViewModel viewModel;
24     private UserLogic userLogic = new UserLogic();
25     private String userName = "testUserName", password = "
testPassword", country = "testCountry", gander = "testGander",
    nameOfTable;
26     private String expenseDate = "01/01/2020", expenseClassification
    = "testClassification", expenseDetails = "testDetails";
27     private double expenseCost = 5.55;
28     private Expense expense = new Expense(expenseDate,
    expenseClassification, expenseDetails, expenseCost);
29
30
31     public ManagementViewModelTest(){
32         LoginForm.username = "testUserName";
33         viewModel = ManagementViewModel.getInstance();
34         viewModel.setExpensesLogic(new ExpensesLogic());
35         nameOfTable = LoginForm.username.toUpperCase() + "_Expenses".
    toUpperCase();
36     }
37
38     @Before
39     public void setUp() {
40         userLogic.addUser(userName, password, country, gander);

```

```
41     }
42
43     @After
44     public void tearDown() throws JDBCDataBaseException, SQLException
45     {
46         while(userLogic.checkIfExist("userName") || userLogic.
47         checkIfExist(userName)) {
48             JDBCDataBase.getInstance().getStatement().executeUpdate("
49             DELETE FROM Registered_Users WHERE UserName = 'testUserName'");
50             JDBCDataBase.getInstance().getStatement().executeUpdate("
51             DELETE FROM Registered_Users WHERE UserName = 'userName'");
52         }
53     }
54
55     @Test
56     public void getInstance() {
57         assertNotNull(ManagementViewModel.getInstance());
58     }
59
60     @Test
61     public void verifyUser() {
62         assertEquals(true, viewModel.verifyUser(userName, password));
63     }
64
65     @Test
66     public void register() {
67         assertEquals(true, viewModel.register("userName", password,
68         country, gander));
69         assertEquals(false, viewModel.register(userName, password,
70         country, gander));
71     }
72
73     @Test
74     public void getAllExpenses() throws GetAllUserExpensesException,
75     SQLException, FormCastingException {
76         ResultSet rs = viewModel.getAllExpenses();
77         int counter = 0;
78
79         while (rs.next()){
80             counter++;
81         }
82         assertEquals(0, counter);
83     }
84 }
```