**Basics of Java Practice Exercise – Day 13**

Submitted By: Aniket Singh (Emp ID: 2605511)

Modify Problem Statement 1 of Day12 to Create ProductManagementTest class to implement all the test cases using JUNIT with respect to CRUD operations. (C – Create, R – Read, U – Update, D - Delete). ProductManagementTest class consist of following members. Refer below steps,

a. Add Connection connection as a Data Member In the class, connection reference required for ProductDAO instance for establishing connection and performing CRUD operations.

b. Add ProductManagementDAO productDAO as a Data Member In the class, productDAO

reference is required to call all the CRUD methods from ProductManagementDAO class.

c. Define public static void init() method and annotate that method with @BeforeClass

annotation, this method calls the static method getConnection() of DBUtil class to get the

connection instance. (This method will be called only once before execution of all test cases).

d. Define public void beforeEachTest() method and annotate that method with @Before

annotation, this method instantiates the ProductManagementDAO class using connection

object and separate instance of ProductManagementDAO available for every test case method. (This method will be called before each test case method).

e. Define public void addProductTest() method and annotate that method with @Test annotation, this method verifies the actual and expected output with respect to insertion of a product in the database.

f. Define public void getAllProductsTest() method and annotate that method with @Test

annotation, this method verifies the actual and expected output with respect to getting all

products from the database.

g. Define public void deleteProductTest() method and annotate that method with @Test

annotation, this method verifies the actual and expected output with respect to deletion of a

product in the database.

h. Define public void updateProductTest() method and annotate that method with @Test

annotation, this method verifies the actual and expected output with respect to

updating a product in the database.

i. Define public static void destroy () method and annotate that method with @AfterClass

annotation, this method calls the closeConnection() method of DBUtil class to close the

Database connection. (This method will be called only once after execution of all test cases)

**Code –**

**package** practiceday13;

**import** java.sql.Connection;

**import** java.sql.SQLException;

**import** java.util.List;

**import** org.junit.\*;

**import** com.mphasis.app.ProductManagementApp;

**import** com.mphasis.dao.ProductManagementDAO;

**import** com.mphasis.dbutil.DBUtil;

**import** com.mphasis.domain.Product;

**public** **class** TestDemo{

**private** **static** Connection *connection*;

**private** ProductManagementDAO dao;

@BeforeClass

**public** **static** **void** init() **throws** Exception {

*connection* = DBUtil.getConnection();

}

@Before

**public** **void** beforeEachTest(){

dao = **new** ProductManagementDAO();

}

@After

**public** **void** tearDown() **throws** Exception {

}

@Test

**public** **void** addProductTest() **throws** SQLException {

Product p=**new** Product("P001", "Mobile", 100);

dao.addProduct(p);

Product newprod = dao.getProductById(P001);

assertNotNull("Product should't be Null",newprod);

assertEquals("Product Name should match", "Mobile", newprod.getName());

assertEquals("Product Price should match", 1000, newprod.getPrice(),0.01);

}

@Test

**public** **void** getAllProductsTest() **throws** SQLException {

List<Product> products = dao.getAllProducts();

assertNotNull("Products List should not be null", products);

assertTrue("Products List should not be empty", products.size()>0);

}

@Test

**public** **void** deleteProductTest() **throws** SQLException {

dao.deleteProduct(P003);

Product delProd = dao.getProductById(P003);

assertNull("Deleted product should be null", delProd);

}

@Test

**public** **void** updateProductTest() **throws** SQLException {

Product p=**new** Product("P002", "Laptop HP", 42000.0);

dao.updateProduct(p);

Product newprod = dao.getProductById(P002);

assertEquals("Product Name should match", "Laptop HP", newprod.getName());

assertEquals("product proce should be match", 42000.0, newprod.getPrice(),0.01);

}

@AfterClass

**public** **static** **void** destroy() {

**if**(*connection* != **null**) {

**try** {

*connection*.close();

}

**catch** (SQLException e) {

e.printStackTrace();

}

}

}

}