Лабораторна робота №11

Тема: Java та бази даних

Мета роботи: набути навичок створення зв'язку Java програми з базою даних та взаємодії з нею в процесі роботи програми .

Хід роботи:

Завдання 1. Створити консольний Java проект java_lab_11 з пакетом com.education.ztu. Необхідно реалізувати програму використовуючи DDL та DML оператори SQL для роботи зі списком товарів.

Завдання 2. Створити з'єднання з базою даних:

• Обрати базу даних з якою буде працювати ваша програма, скачати для неї JDBC драйвер та додати в проект.

• Створити базу даних store.

• Налаштувати з'єднання (параметри з'єднання повинні бути збережені в properties файлі та зчитуватись з ResourceBundle).

Лістинг програми:

DatabaseConnection.java:

```
package com.education.ztu;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class DatabaseConnection {

    private static final ResourceBundle bundle = ResourceBundle.getBundle("db");

    public static Connection getConnection() throws SQLException {
        try {
            String url = bundle.getString("db.url");
            String username = bundle.getString("db.username");
            String password = bundle.getString("db.password");
            String driverClass = bundle.getString("db.driverClass");
            Class.forName(driverClass);

            return DriverManager.getConnection(url, username, password);
        } catch (ClassNotFoundException e) {
            throw new SQLException("JDBC Driver не знайдено.", e);
        }
    }
}
```

Завдання 3. Робота з ба зою даних використовуючи клас Statement:

• Створити необхідні таблицю чи таблиці в базі даних

• Заповнити їх даними для 10 товарів (тут краще використати batchкоманди)

• Отримати всі записи з бази з інформацією про товари та вивести їх в консоль.

Завдання 4. Робота з базою даних використовуючи клас PreparedStatement:

• Додати ще 5 товарів використовуючи.

• Отримати дані про товари з певної категорії чи певного бренду та вивести їх в консоль.

• Після цього видалити всі записи з бази.

Завдання 5. Робота з транзакціями та точками збереження: Додати два товари (один запит повинен бути з синтаксичними помилками) Створити точку збереження після додавання першого товару. Вивести в консоль товари, що були додані після відпрацювання двох запитів.

Лістинг програми:

```
package com.education.ztu;

import java.sql.*;

public class ProductManager {
    public static void getAllProducts() {
        try (Connection connection = DatabaseConnection.getConnection();
             Statement statement = connection.createStatement()) {

            String selectSQL = "SELECT * FROM products";
            ResultSet resultSet = statement.executeQuery(selectSQL);

            System.out.println("Список товарів:");
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                double price = resultSet.getDouble("price");
                int quantity = resultSet.getInt("quantity");

                System.out.printf("ID: %d, Назва: %s, Ціна: %.2f, Кількість:
%d%n", id, name, price, quantity);
            }

        } catch (SQLException e) {
            System.err.println("Помилка при отриманні даних з таблиці.");
            e.printStackTrace();
        }
    }

    public static void getProductsByPrice(double price, String condition) {
        String selectSQL = "SELECT * FROM products WHERE price " + condition + "
?";
```

```
        try (Connection connection = DatabaseConnection.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(selectSQL)) {

            preparedStatement.setDouble(1, price);

            ResultSet resultSet = preparedStatement.executeQuery();
            System.out.println("Товари з ціною " + condition + " " + price +
":");

            boolean hasProducts = false;

            while (resultSet.next()) {
                hasProducts = true;
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                double productPrice = resultSet.getDouble("price");
                int quantity = resultSet.getInt("quantity");

                System.out.printf("ID: %d, Назва: %s, Ціна: %.2f, Кількість:
%d%n", id, name, productPrice, quantity);
            }

            if (!hasProducts) {
                System.out.println("Товари не знайдено.");
            }

        } catch (SQLException e) {
            System.err.println("Помилка при отриманні товарів за ціною.");
            e.printStackTrace();
        }
    }

    public static void addProductsWithTransaction() {
        String insertProduct1 = "INSERT INTO products (name, price, quantity)
VALUES ('Товар 1', 20.50, 10)";
        String insertProduct2WithError = "INSERT INTO products (name, price,
quantiy) VALUES ('Товар 2', 15.00, 5)";

        try (Connection connection = DatabaseConnection.getConnection()) {
            connection.setAutoCommit(false);

            try (PreparedStatement statement1 =
connection.prepareStatement(insertProduct1);
                 PreparedStatement statement2 =
connection.prepareStatement(insertProduct2WithError)) {

                statement1.executeUpdate();
                System.out.println("Перший товар успішно додано.");

                connection.setSavepoint("Savepoint1");
                System.out.println("Точка збереження створена.");

                statement2.executeUpdate();
                System.out.println("Другий товар успішно додано.");

                connection.commit();
            } catch (SQLException e) {
                System.err.println("Помилка при додаванні товарів: " +
e.getMessage());
                System.out.println("Відновлення до точки збереження.");
                connection.rollback(connection.setSavepoint("Savepoint1"));
                connection.commit();
            }
```

```
        } catch (SQLException e) {
            System.err.println("Помилка транзакції: " + e.getMessage());
        }
    }
}
```

```
package com.education.ztu;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;

public class DatabaseSetup {
    public static void createTableAndInsertData() {
        try (Connection connection = DatabaseConnection.getConnection();
             Statement statement = connection.createStatement()) {

            String createTableSQL = """
                    CREATE TABLE IF NOT EXISTS products (
                        id INT AUTO_INCREMENT PRIMARY KEY,
                        name VARCHAR(100) NOT NULL,
                        price DECIMAL(10, 2) NOT NULL,
                        quantity INT NOT NULL
                    );
                    """;
            statement.execute(createTableSQL);
            System.out.println("Таблиця 'products' створена.");

            String insertDataSQL = """
                    INSERT INTO products (name, price, quantity) VALUES
                    ('Milk', 35.50, 100),
                    ('Chocolate', 15.50, 200),
                    ('Juice', 9.75, 150),
                    ('Orange', 25.00, 80),
                    ('Apple', 28.20, 300),
                    ('Cheese', 45.99, 50),
                    ('Butter', 34.25, 10),
                    ('Tea', 40.00, 75),
                    ('Tomato', 68.50, 60),
                    ('Cake', 120.00, 180);
                    """;
            statement.addBatch(insertDataSQL);
            statement.executeBatch();

            System.out.println("Дані успішно додані до таблиці 'products'.");

        } catch (SQLException e) {
            System.err.println("Помилка при створенні таблиці або додаванні
даних.");
            e.printStackTrace();
        }
    }

    public static void addMoreProducts() {
        String insertSQL = "INSERT INTO products (name, price, quantity) VALUES
(?, ?, ?)";

        try (Connection connection = DatabaseConnection.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(insertSQL)) {
            connection.setAutoCommit(false);

            preparedStatement.setString(1, "Pudding");
            preparedStatement.setDouble(2, 25.50);
```

```
            preparedStatement.setInt(3, 12);
            preparedStatement.addBatch();

            preparedStatement.setString(1, "Coffee");
            preparedStatement.setDouble(2, 14.00);
            preparedStatement.setInt(3, 5);
            preparedStatement.addBatch();

            preparedStatement.setString(1, "Товар 13");
            preparedStatement.setDouble(2, 9.99);
            preparedStatement.setInt(3, 150);
            preparedStatement.addBatch();

            preparedStatement.setString(1, "Ice cream");
            preparedStatement.setDouble(2, 18.25);
            preparedStatement.setInt(3, 50);
            preparedStatement.addBatch();

            preparedStatement.setString(1, "Feta");
            preparedStatement.setDouble(2, 85.75);
            preparedStatement.setInt(3, 30);
            preparedStatement.addBatch();

            preparedStatement.executeBatch();
            connection.commit();
            System.out.println("Додано ще 5 товарів у таблицю 'products'!");

        } catch (SQLException e) {
            System.err.println("Помилка при додаванні товарів.");
            e.printStackTrace();
        }
    }
}
```

```
package com.education.ztu;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class DatabaseCleanup {
    public static void deleteAllProducts() {
        String deleteSQL = "DELETE FROM products";

        try (Connection connection = DatabaseConnection.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(deleteSQL)) {

            int rowsAffected = preparedStatement.executeUpdate();
            System.out.println("Видалено записів: " + rowsAffected);

        } catch (SQLException e) {
            System.err.println("Помилка при видаленні записів.");
            e.printStackTrace();
        }
    }
}
```

```
package com.education.ztu;
import java.sql.Connection;
import java.sql.SQLException;

public class Main {
    public static void main(String[] args) {
```

```
        try (Connection connection = DatabaseConnect ion.getConnection()) {
            if (connection != null) {
                System.out.println("З'єднання з базою даних успішне!");

                DatabaseSetup.createTableAndInsertData();

                ProductManager.getAllProducts();
                ProductManager.getProductsByPrice(30.0, "<");
                DatabaseCleanup.deleteAllProducts();
                ProductManager.addProductsWithTransaction();
                ProductManager.getProductsByPrice(0, ">");
            }
        } catch (SQLException e) {
            System.err.println("Помилка з'єднання з базою даних.");
            e.printStackTrace();
        }
    }
}
```

Результат виконання програми:

```
З'єднання з базою даних успішне!
Таблиця 'products' створена.
Дані успішно додані до таблиці 'products'.
Список товарів:
ID: 1, Назва: Milk, Ціна: 35,50, Кількість: 100
ID: 2, Назва: Chocolate, Ціна: 15,50, Кількість: 200
ID: 3, Назва: Juice, Ціна: 9,75, Кількість: 150
ID: 4, Назва: Orange, Ціна: 25,00, Кількість: 80
ID: 5, Назва: Apple, Ціна: 28,20, Кількість: 300
ID: 6, Назва: Cheese, Ціна: 45,99, Кількість: 50
ID: 7, Назва: Butter, Ціна: 34,25, Кількість: 10
ID: 8, Назва: Tea, Ціна: 40,00, Кількість: 75
ID: 9, Назва: Tomato, Ціна: 68,50, Кількість: 60
ID: 10, Назва: Cake, Ціна: 120,00, Кількість: 180
```

```
Товари з ціною < 30.0:
ID: 2, Назва: Chocolate, Ціна: 15,50, Кількість: 200
ID: 3, Назва: Juice, Ціна: 9,75, Кількість: 150
ID: 4, Назва: Orange, Ціна: 25,00, Кількість: 80
ID: 5, Назва: Apple, Ціна: 28,20, Кількість: 300
ID: 12, Назва: Chocolate, Ціна: 15,50, Кількість: 200
ID: 13, Назва: Juice, Ціна: 9,75, Кількість: 150
ID: 14, Назва: Orange, Ціна: 25,00, Кількість: 80
ID: 15, Назва: Apple, Ціна: 28,20, Кількість: 300
Видалено записів: 20
Перший товар успішно додано.
Точка збереження створена.
Відновлення до точки збереження.
Помилка при додаванні товарів: Unknown column 'quantiy' in 'field list'
Товари з ціною > 0.0:
ID: 21, Назва: Товар 1, Ціна: 20,50, Кількість: 10
```

Завдання 6. Реалізація взаємодії з базою даних товарів використовуючи DAO: Створити абстрактний клас AbstractDAO з абстрактними методами для

подальшої реалізації CRUD операцій Реалізувати ProductDAO, що наслідується від AbstractDAO та імплементувати необхідні методи. В тестовому класі продемонструвати взаємодію з базою даних використовуючи ProductDAO.

Лістинг програми:

```
package com.education.ztu.Task6;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

public abstract class AbstractDAO<T> {
    protected Connection connection;

    public AbstractDAO(Connection connection) {
        this.connection = connection;
    }
    public abstract void create(T entity) throws SQLException;
    public abstract T read(int id) throws SQLException;
    public abstract void update(T entity) throws SQLException;
    public abstract void delete(int id) throws SQLException;
    public abstract List<T> getAll() throws SQLException;
}
```

```
package com.education.ztu.Task6;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO extends AbstractDAO<Product> {

    public ProductDAO(Connection connection) {
        super(connection);
    }

    @Override
    public void create(Product product) throws SQLException {
        String sql = "INSERT INTO products (name, price, quantity) VALUES (?, ?,
?)";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setString(1, product.getName());
            preparedStatement.setDouble(2, product.getPrice());
            preparedStatement.setInt(3, product.getQuantity());
            preparedStatement.executeUpdate();
        }
    }

    @Override
    public Product read(int id) throws SQLException {
        String sql = "SELECT * FROM products WHERE id = ?";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setInt(1, id);
            ResultSet resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                return new Product(
                        resultSet.getInt("id"),
                        resultSet.getString("name"),
                        resultSet.getDouble("price"),
                        resultSet.getInt("quantity"),
```

```
                );
            }
            return null;
        }
    }

    @Override
    public void update(Product product) throws SQLException {
        String sql = "UPDATE products SET name = ?, price = ?, quantity = ?
WHERE id = ?";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setString(1, product.getName());
            preparedStatement.setDouble(2, product.getPrice());
            preparedStatement.setInt(3, product.getQuantity());
            preparedStatement.setInt(4, product.getId());
            preparedStatement.executeUpdate();
        }
    }

    @Override
    public void delete(int id) throws SQLException {
        String sql = "DELETE FROM products WHERE id = ?";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setInt(1, id);
            preparedStatement.executeUpdate();
        }
    }

    @Override
    public List<Product> getAll() throws SQLException {
        List<Product> products = new ArrayList<>();
        String sql = "SELECT * FROM products";
        try (Statement statement = connection.createStatement();
             ResultSet resultSet = statement.executeQuery(sql)) {
            while (resultSet.next()) {
                products.add(new Product(
                        resultSet.getInt("id"),
                        resultSet.getString("name"),
                        resultSet.getDouble("price"),
                        resultSet.getInt("quantity")
                ));
            }
        }
        return products;
    }
}
```

```
package com.education.ztu.Task6;

public class Product {
    private int id;
    private String name;
    private double price;
    private int quantity;

    public Product(int id, String name, double price, int quantity) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public int getId() {
```

```
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Product{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", price=" + price +
                ", quantity=" + quantity +
                '}';
    }
}
```

```
package com.education.ztu.Task6;

import com.education.ztu.DatabaseConnection;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        try (Connection connection = DatabaseConnection.getConnection()) {
            if (connection != null) {
                System.out.println("З'єднання з базою даних успішне!");
                ProductDAO productDAO = new ProductDAO(connection);

                Product newProduct = new Product(0, "Товар 1", 100.50, 20);
                productDAO.create(newProduct);
                System.out.println("Товар додано: " + newProduct);

                List<Product> products = productDAO.getAll();
                System.out.println("Всі товари:");
                for (Product product : products) {
                    System.out.println(product);
```

```
                }

                newProduct.setPrice(110.00);
                productDAO.update(newProduct);
                System.out.println("Оновлений товар: " + newProduct);

                productDAO.delete(newProduct.getId());
                System.out.println("Товар з ID " + newProduct.getId() + "
видалено.");
            }
        } catch (SQLException e) {
            System.err.println("Помилка з'єднання з базою даних.");
            e.printStackTrace();
        }
    }
}
```

Результат виконання програми:

```
Товар додано: Product{id=0, name='Товар 1', price=100.5, quantity=20}
Всі товари:
Product{id=21, name='Товар 1', price=20.5, quantity=10}
Product{id=22, name='Товар 1', price=100.5, quantity=20}
Оновлений товар: Product{id=0, name='Товар 1', price=110.0, quantity=20}
Товар з ID 0 видалено.
```

Висновок: у результаті виконання лабораторної роботи я отримала навички створення зв'язку Java програми з базою даних та взаємодії з нею в процесі роботи програми .