

## Лабораторна робота №9. Простий Java клієнт до бази даних

**Мета роботи:** Отримання практичних навичок створення простого клієнту до бази даних (4 години)

### Завдання

Розробити віконний додаток, в якому буде відображатися інформація з бази даних. Крім відображення інформації з БД, він повинен дозволяти додавати та видаляти записи. Дані представити у вигляді таблиць (використовувати компонент JTable).

Складність БД - необмежена. Мінімум - вона повинна містити хоча б одну таблицю.

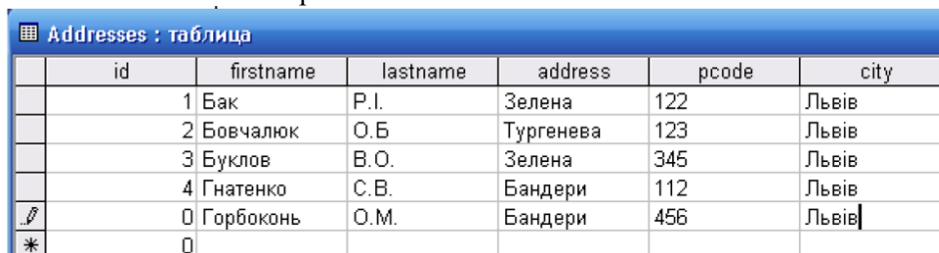
*Виберіть та виконайте варіант, який співпадає з Вашим номером у списку*

1. Продуктовий магазин
2. Автовокзал
3. Автосалон
4. Аптека
5. Банк
6. Банкетний зал
7. Виборча дільниця
8. Відділ кадрів
9. Журнал олімпіади
10. Поштове відділення
11. Магазин побутової хімії
12. Магазин одягу
13. Журнал студентів
14. Кінотеатр
15. Каталог навчальних курсів

### Теоретичні відомості

#### Створення БД

Створіть базу даних за допомогою програми MS Access з іменем Romanchuk. В БД створіть таблицю з іменем Addresses з полями id, firstname, lastname, address, pcode, city та введіть п'ять записів як показано на рис. 8.1 :



	id	firstname	lastname	address	pcode	city
	1	Бак	P.I.	Зелена	122	Львів
	2	Бовчалюк	O.B.	Тургенева	123	Львів
	3	Буклов	B.O.	Зелена	345	Львів
	4	Гнатенко	C.B.	Бандери	112	Львів
	5	Горбоконь	O.M.	Бандери	456	Львів

Рис. 8.1. Тестова таблиця

#### Налаштування ODBC.

Для цього спочатку настроїмо баз даних та операційну систему для того, щоб забезпечити доступ до бази із нашої Java-програми з допомогою ODBC (Open Database Connectivity) мосту. Для цього необхідно зробити наступні дії:

Заходимо в меню Control Panel -> Administrative Tools. (Для windows 7 c:\windows\SysWOW64\odbcsrc32.exe). Запускаємо програму Data Sources (ODBC), у вікні програми

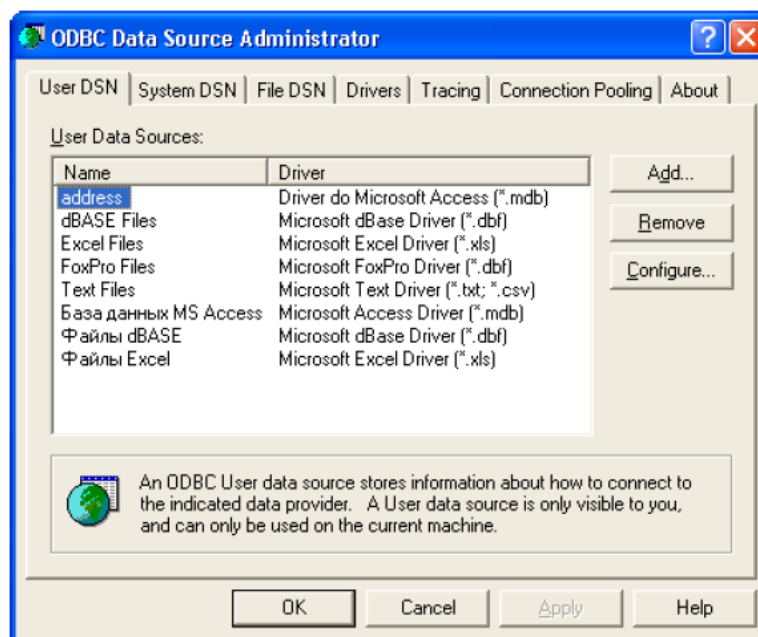


Рис. 8.2 Натискаємо кнопку Add...

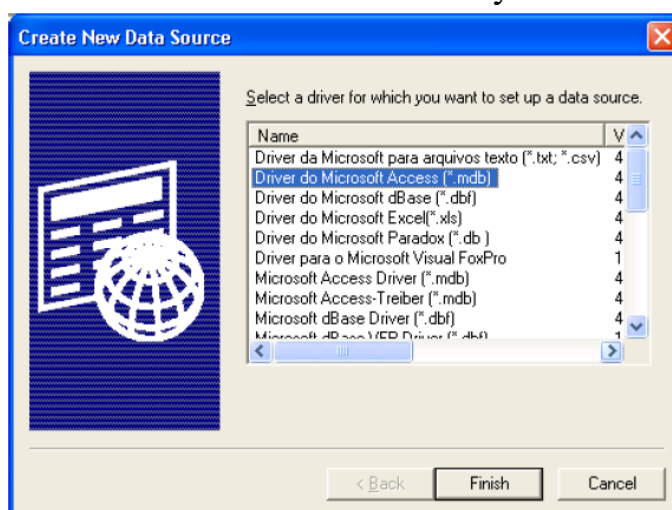


Рис.8.3. В даному вікні вибираємо Driver do MicrosoftAccess(\*.mdb), натискаємо кнопку Finish.

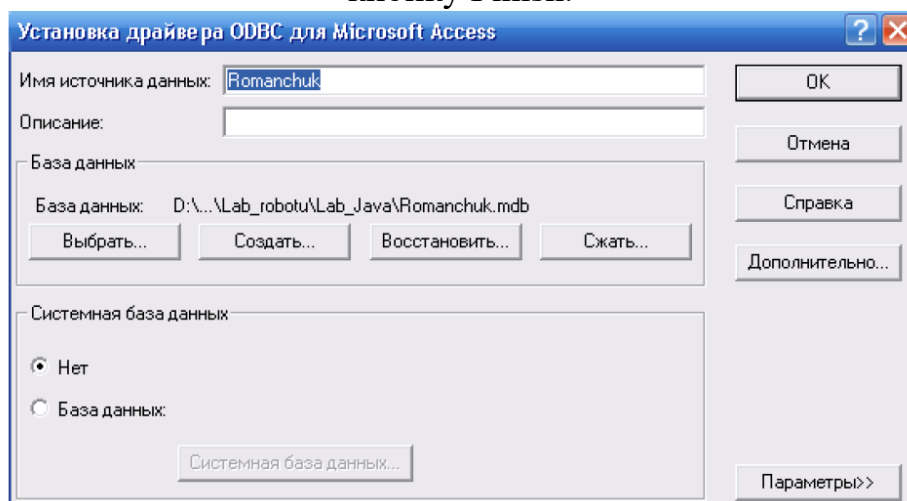


Рис. 8.4. В даному вікні натисніть кнопку Select... і вкажіть файл бази даних – Адресної книжки

Введіть Data Source Name, натисніть ОК. У вікні ODBC Data Source Administrator повинен з'явитись рядок із назвою підключеної бази даних. Натисніть ОК. (При цьому в Microsoft Access вищезгадана база даних повинна бути закритою).

Приклад.

```
import javax.swing.*;
import javax.swing.table.AbstractTableModel;
import java.sql.*;
import java.util.List;
import java.util.ArrayList;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class Lab8 extends JFrame
{
    private Connection connection;
    private Statement statement;
    private UserData userData;
    public Lab8()
    {
        initConnection();
        buildUI();
        loadDataFromDB();
        this.userData.fireTableDataChanged();
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                try
                {
                    statement.close();
                    connection.close();
                }
                catch (SQLException e1)
                {
                    e1.printStackTrace();
                }
                System.exit(0);
            }
        });
        setVisible(true);
    }
    private void initConnection()
    {
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            connection=DriverManager.getConnection("jdbc:odbc:Romanchuk");
            statement = connection.createStatement();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```

private void buildUI()
{
    setSize(400, 400);
    setTitle("Address Book");
    getContentPane().setLayout(new BorderLayout());
    this.userData = new UserData();
    JTable userTable = new JTable(userData);
    getContentPane().add(new JScrollPane(userTable),
BorderLayout.CENTER);
}
private void loadDataFromDB()
{
    this.userData.alData.clear();
    try
    {
        ResultSet resultSet = statement.executeQuery("SELECT *
FROM Addresses");
        while (resultSet.next())
        {
            String id = resultSet.getString(1);
            String firstname = resultSet.getString(2);
            String lastname = resultSet.getString(3);
            String address = resultSet.getString(4);
            String city = resultSet.getString(5);
            RowAddress r = new RowAddress(id, firstname,
lastname, address, city);
            this.userData.alData.add(r);
        }
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
class RowAddress
{
    String id;
    String firstname;
    String lastname;
    String address;
    String city;
    public RowAddress(String id, String firstname, String
lastname, String address, String city)
    {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.address = address;
        this.city = city;
    }
}
class UserData extends AbstractTableModel
{
    String columns[] = {"First Name", "Last Name"};
    public List alData = new ArrayList();
    public int getColumnCount()
    {
        return columns.length;
    }
    public Object getValueAt(int nRow, int nCol)

```

```

        {
            if (nRow < 0 || nRow > this.alData.size())
            {
                return null;
            }
            RowAddress rowAddress = (RowAddress)
this.alData.get(nRow);
            switch (nCol)
            {
                case 0: return rowAddress.firstname;
                case 1: return rowAddress.lastname;
            }
            return "";
        }
        public int getRowCount()
        {
            return alData == null ? 0 : alData.size();
        }
        public String getColumnName(int column)
        {
            return columns[column];
        }
        public boolean isCellEditable(int nRow, int nCol)
        {
            return false;
        }
    }
    public static void main(String[] args)
    {
        new Lab8();
    }
}

```

**Пояснення до прикладу:**

Даний клас Lab8 є підкласом **JFrame** і служить основним вікном для нашого прикладу. Змінні(поля) класу:

**private Connection connection** - служить для підключення до бази даних;

**private Statement statement** - служить для виконання SQL-команд;

**private UserData userData** - для зберігання результатів у вигляді таблиці.

В конструкторі викликаються 3 основні методи:

`initConnection();`

`buildUI();`

`loadDataFromDB();`

Вся процедура підключення до бази даних здійснюється в першій функції:

`Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`

`connection =`

`DriverManager.getConnection("jdbc:odbc:syrotynskyAB");`

`statement = connection.createStatement();`

При цьому, спочатку нам необхідно завантажити драйвер-міст

**jdbc-odbc**, а потім відбувається підключення до нашої бази даних.

Назва бази даних, тобто тієї бази, котру ми ввели в вікні,

показаному на Рис.8.3, вводиться як аргумент функції

`DriverManager.getConnection("jdbc:odbc:Romanchuk").`

Якщо назва бази даних введена нами неправильно, тоді виникає

виняткова ситуація - виконається функція в блоці catch().

Для безпосереднього виконання команд SQL необхідно створити об'єкт класу Statement. Цей об'єкт служить для керування транзакціями. По замовчуванню, для нового об'єкту задається режим автоматичної фіксації, тобто результати виконання кожної команди відразу фіксуються в базі даних.

Після виконання підключення до бази, виконується функція buildUI(). В цій функції задаються основні параметри вікна та створюється таблиця, котра потім додається до вікна. Існує декілька конструкторів класу Jtable. В нашому випадку, таблиця створюється на основі підкласу AbstractTableModel - UserData. В цьому класі є декілька функцій, котрі визначають вигляд таблиці. Найважливіша функція:

```
public Object getValueAt(int nRow, int nCol)
{
    if (nRow < 0 || nRow > this.alData.size())
    {
        return null;
    }
    RowAddress rowAddress = (RowAddress) this.alData.get(nRow);
    switch (nCol)
    {
        case 0: return
            rowAddress.firstname;
        case 1: return
            rowAddress.lastname;
    }
    return "";
}
```

Тут задаються параметри, котрі будуть відображатись у відповідних рядках та колонках таблиці. Для того, щоб локально зберігати результат виконання SQL запиту, ми створили внутрішній клас RowAddress. В цьому класі є лише декілька полів, в котрих зберігаються відповідні комірки бази даних.

Отримання результатів із бази даних здійснюється в функції loadDataFromDB(). Для цього спочатку виконується SQL запит: statement.executeQuery(" SELECT \* FROM Addresses"). Результатом виконання цього запиту є об'єкт класу ResultSet. В цьому об'єкті колонки зберігається в тому ж порядку, що і в таблиці бази даних. Для отримання значень відповідних колонок використовується метод resultSet.getString(l) із номером колонки. Потім новостворений об'єкт RowAddress додається до динамічного масиву в userData.

Після виконання SQL запиту в конструкторі Lab7 викликається метод this.userData.fireTableDataChanged() - виконання цього методу необхідне для того, щоб правильно відтворити таблицю із отриманими результатами.

Для того, щоб коректно завершити програму, нам необхідно від'єднатись від бази даних. Для цього виконуються методи: statement.close(); connection.close().

2. За що відповідає номера (1,2,3,4,5,6) в команда String id = resultSet.getString(1); ... ,даної програми?

```
private void loadDataFromDB()
{
    this.userData.alData. clear(); try
    {
        ResultSet resultSet = statement.executeQuery("SELECT * FROM
Addresses"); while (resultSet.next())
        {
            String id = resultSet.getString(1);
            String firstname = resultSet.getString(2);
            String lastname = resultSet.getString(3);
            String address = resultSet.getString(4);
            String city = resultSet.getString(5);
            RowAddress r = new RowAddress(id, firstname, lastname, address, city);
            this.userData.alData.add(r);
        }
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

3. Яка з цих команд відповідає за з'єднання з БД? try

```
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    connection = DriverManager.getConnection("jdbc:odbc:romanchuk");
    statement = connection.createStatement();
}

    catch (Exception e)
{
    e. printStackTrace() ;
}
```

4. Яка з цих команд відповідає за підключення драйвера JDBC-ODBC? try

```
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    connection = DriverManager.getConnection("jdbc:odbc:romanchuk");
    statement = connection.createStatement();
}

    catch (Exception e)
{
    e. printStackTrace() ;
}
```

5. Який драйвер використовується<sup>7</sup> для підключення БД в адміністративній консолі Windows

6. Який драйвер використовується для підключення до БД з Java-програми?