NewJFrame.java

```
public class NewJFrame extends javax.swing.JFrame {
Connection con;
    /**
     * Creates new form NewJFrame
     */
    public NewJFrame() {
        initComponents();
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/books", "eugeny", "123");

        } catch (SQLException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
        }

    }


    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        //(прокрутка ,или  CONCUR_UPDATABLE(чтобы редактировать записи))
        Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = st.executeQuery("SELECT * FROM author");
        jTable1.setModel(new MyTableModel(rs));
    } catch (SQLException ex) {
        Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
}
```

`AbstractTableModel`
Этот абстрактный класс обеспечивает реализацию по умолчанию для большинства методов в интерфейсе `TableModel`. Он заботится об управлении слушателями и обеспечивает некоторые удобства для того, чтобы они генерировали `TableModelEvents` и обеспечивали диспетчеризацию событий слушателям. Создав класс `TableModel` как подкласс `AbstractTableModel`, вы должны только обеспечить реализацию для следующих трех методов:

```
  public int getRowCount();
  public int getColumnCount();
  public Object getValueAt(int row, int column);
```

Класс ResultSet представляет результирующий набор данных и обеспечивает приложению построчный доступ к результатам запросов. При обработке запроса *ResultSet* поддерживает указатель на текущую обрабатываемую строку.

Доступ к данным **ResultSet** обеспечивает посредством набора get-методов, которые организуют доступ к колонкам текущей строки. Метод ResultSet.next используется для перемещения к следующей строке ResultSet, делая ее текущей.

MyTableModel.java

```
package qwer;

class MyTableModel extends AbstractTableModel {
    String[] headers;
    ResultSet rs;
    public MyTableModel(ResultSet rs) {
        this.rs = rs;
        try {
            //Имена колонок и какого они типа - можно достать
            ResultSetMetaData metaData = rs.getMetaData();
            int columnCount = metaData.getColumnCount();
            headers = new String[columnCount];
            for (int i = 0; i < headers.length; i++) {
                headers[i]= metaData.getColumnName(i+1);
            }
        } catch (SQLException ex) {
            Logger.getLogger(MyTableModel.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    public int getRowCount() {
        try {
            //сколько строк  4            //дойти до последнего и спросить: где я?
            rs.last();
            return rs.getRow();
        } catch (SQLException ex) {
            return 0;
        }
    }

    @Override
    public int getColumnCount() {
        return headers.length;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {// 5
        try {
            rs.absolute(rowIndex+1);//поставить на нужную строчку
            return rs.getObject(columnIndex+1);
        } catch (SQLException ex) {
            return " ";
        }
    }

    @Override
    public String getColumnName(int column) {
        return headers[column];
    }

    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return columnIndex>0;
    }


}
```