

Лекция 8

Работа с базами данных



Введение

База данных состоит из хранимых данных и системы управления базой данных (СУБД).

Наиболее часто на практике используются так называемые **реляционные базы данных**.

Данные хранятся в виде таблиц, у которых имеется фиксированное число столбцов.

Каждый столбец имеет свое уникальное имя.

Каждая строка таблицы называется **ЗАПИСЬЮ**.

Запись состоит из отдельных **ПОЛЕЙ** (столбцы таблицы).

Фрагмент базы данных аэрологического зондирования атмосферы AERO

data

station	time	height	latitude	longitude	p	t	e	direction	v
26063	2009-01-01 12:00	2	59.95	30.70	99800	266.85	286.36	300	7
26063	2009-01-01 12:00	598	59.95	30.70	92500	262.45	216.69	330	12.5
26063	2009-01-01 12:00	648	59.95	30.70	91900	261.65	203.29	330	12.5
26063	2009-01-01 12:00	767	59.95	30.70	90500	261.65	205.84	330	12.5
26063	2009-01-01 12:00	922	59.95	30.70	88700	262.45	211.28	330	12.5
26063	2009-01-01 12:00	1097	59.95	30.70	86700	260.85	140.60	330	12.5

База данных **aero** содержит таблицу **data**, состоящую из 10 полей и нескольких миллионов записей (строк).

Взаимодействие с базой данных

Для **записи** информации в базу данных и для **чтения** информации из базы данных Java-приложение взаимодействует с СУБД посредством команд на языке SQL (Structured Query Language)



Основные SQL-команды (1)

Для получения данных из БД используется команда **SELECT**. Например,

```
SELECT * FROM data;
```

Если нужно получить информацию только по отдельным полям, например, данные о высоте над земной поверхностью и температуре воздуха, то запрос будет иметь вид

```
SELECT height, t FROM data;
```

Основные SQL-команды (2)

В запросе можно указать номер аэрологической станции, чтобы получить информацию для нужного пункта

```
SELECT height, t FROM data WHERE station=26063;
```

Также целесообразно указать и дату проведения измерений

```
SELECT height, t FROM data WHERE station=26063  
AND time = «2009-01-01 12:00:00»;
```

Создание Java-класса для чтения информации из базы данных (1)

```
import java.sql.*;

public class DataSQL {

    public DataSQL() {
    }

    public Air[] getData(int station, int year, int month, int day, int hour) {
        String dataBaseUrl = "jdbc:mysql://meteolab.ru:3306/aero";
        String dataBaseUser = "student";
        String dataBasePassword = "****_****_****";
        String query = null;
        double z;
        double T;

        // Проверяем наличие драйвера MySQL
        try {
            Class.forName("com.mysql.jdbc.Driver");
        }
        catch(Exception ex) {
            ex.printStackTrace();
            return null;
        }

        // Продолжение следует...
```

Создание Java-класса для чтения информации из базы данных (2)

```
// Читаем данные из базы данных
try {
    // Соединяемся с базой данных
    Connection c =
        DriverManager.getConnection(dataBaseUrl, dataBaseUser, dataBasePassword);
    Statement s =
        c.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

    // Выполняем запрос количества данных
    query = "select distinct height, t from data where station = "+station+" and "+
        "time = \"+year+"-"+month+"-"+day+" "+hour+"\"";
    ResultSet rs = s.executeQuery(query);

    // Подсчитываем число слоев по высоте
    int N = 0;
    while(rs.next()) {
        N++;
    }

    // Создаем массив объектов класса Air
    Air[] air = new Air[N];

    // Продолжение следует...
```


Создание Java-класса для чтения информации из базы данных (3)

```
// Выполняем заново этот же запрос к базе данных
rs = s.executeQuery(query);

// Обрабатываем ответ от базы данных
int i = 0;
while(rs.next()) {
    z = Double.parseDouble(rs.getString(1).trim());
    T = Double.parseDouble(rs.getString(2).trim());
    air[i] = new Air(z, T);
    i++;
}

// Закрываем соединение с базой данных
rs.close();
s.close();
c.close();

// Возвращаем массив объектов типа Air
return air;
}
catch(Exception ex) {
    ex.printStackTrace();
    return null;
}
}
```

Обработчик события нажатия на кнопку (1)

The screenshot shows an IDE window with the following components:

- Projects View:** Shows a project named 'desktopapplication3' with sub-packages 'resources' and 'resources.bu'. Source files include 'Air.java', 'DataFile.java', 'DataSQL.java', 'DesktopApplication3.java', 'DesktopApplication3AboutBox.j', 'DesktopApplication3View.java', and 'Forecast.java'.
- Members View:** Lists members of the 'messageTimer' class, including 'JButton2 : JButton', 'JLabel1 : JLabel', 'JLabel2 : JLabel', 'JLabel3 : JLabel', 'JLabel4 : JLabel', 'JLabel5 : JLabel', 'JLabel6 : JLabel', 'JLabel7 : JLabel', 'JLabel8 : JLabel', 'JMenuItem1 : JMenuItem', 'mainPanel : JPanel', 'menuBar : JMenuBar', and 'messageTimer : Timer'.
- Source Code:** The main editor displays the following Java code:

```
        str = String.format(Locale.US, "%.1f", air.getHeight());
        jLabel2.setText(str);
        str = String.format(Locale.US, "%.1f", air.getSaturatedVapourPressure());
        jLabel3.setText(str);
    }

    @Action
    public void readFile() {
        DataSQL ds = new DataSQL();
        Air[] air = ds.getData(26063, 2009, 1, 1, 12);
        Forecast fc = new Forecast(air, 2000.0);
        String str = String.format(Locale.US, "%.1f", fc.getTmax());
        jLabel7.setText(str);
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JPanel mainPanel;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JProgressBar progressBar;
    private javax.swing.JLabel statusAnimationLabel;
    private javax.swing.JLabel statusMessageLabel;
    private javax.swing.JPanel statusPanel;

    // End of variables declaration

    private final Timer messageTimer;
    private final Timer busyIconTimer;
    private final Icon idleIcon;
```

Обработчик события нажатия на кнопку (2)

```
public void readFile() {  
    DataSQL ds = new DataSQL();  
    Air[] air = ds.getData(26063, 2009, 1, 1, 12);  
    Forecast fc = new Forecast(air, 2000.0);  
    String str = String.format(Locale.US, "%.1f", fc.getTmax());  
    jLabel7.setText(str);  
}
```

Линейная интерполяция температуры между уровнями

Формула для линейной интерполяции температуры
между уровнями z_1 и z_2 :

$$T = T_{i-1} + \frac{T_i - T_{i-1}}{z_i - z_{i-1}} \cdot (H - z_{i-1})$$

Обновленный вариант класса для прогноза максимальной температуры

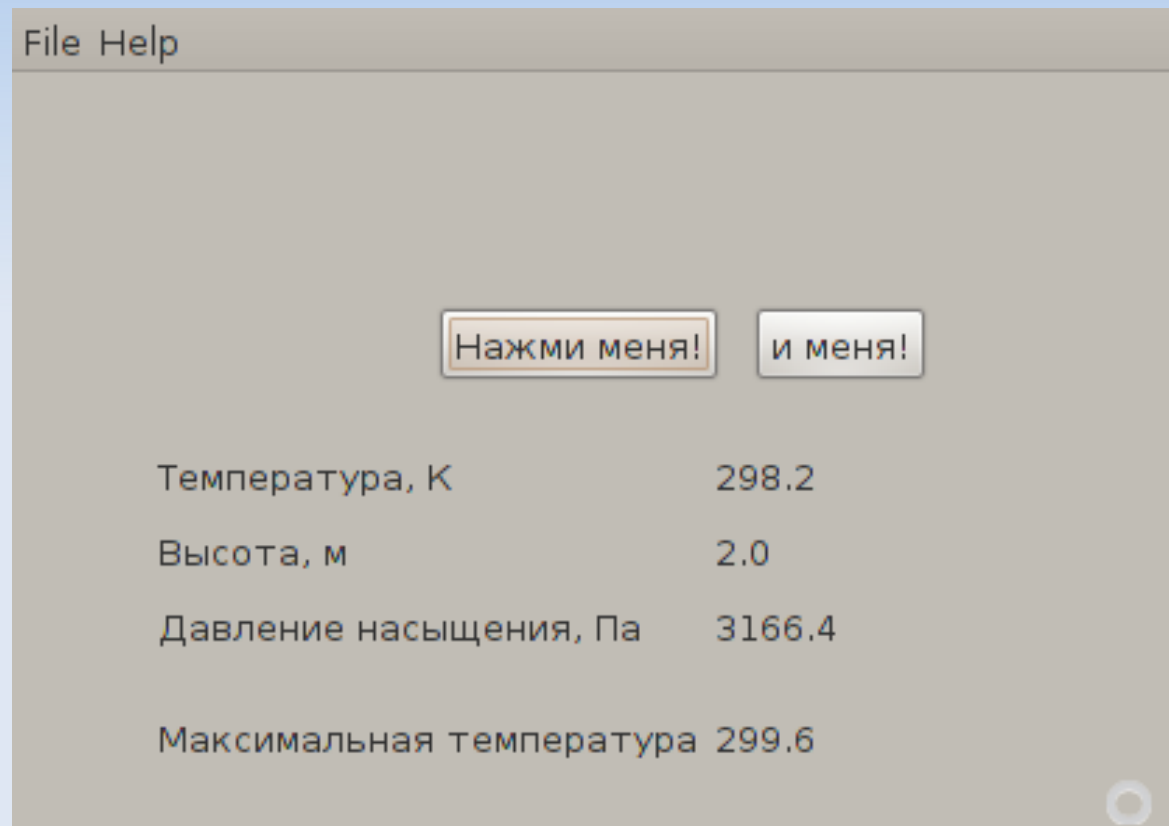
```
import static java.lang.Math.*;

public class Forecast {
    private Air[] air;
    private double H;

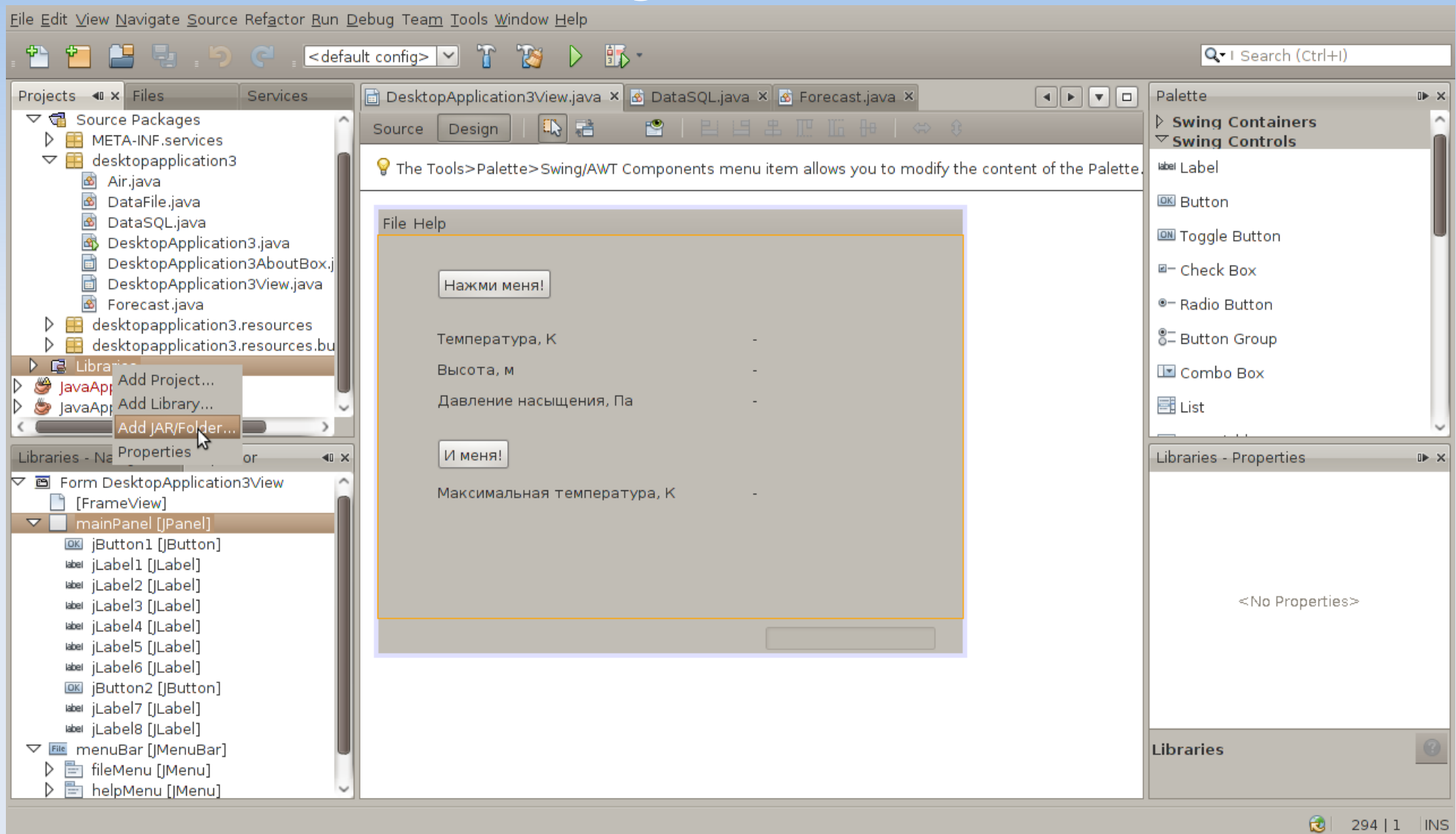
    public Forecast(Air[] air, double H) {
        this.air = air;
        this.H = H;
    }

    public double getTmax() {
        int i;
        double T;
        double Tmax = 0.0;
        for(i=0; i<air.length; i++) {
            if(air[i].getHeight() > H) {
                T = air[i-1].getTemperature() + (air[i].getTemperature()-air[i-1].getTemperature())/
                    (air[i].getHeight()-air[i-1].getHeight())*(H-air[i-1].getHeight());
                Tmax = T + 0.0098*H;
                return Tmax;
            }
        }
        return 0.0;
    }
}
```

Результат работы программы



Подключение драйвера MySQL к проекту в NetBeans



Выбрать пункт **Add JAR/Folder...**

Затем указать путь к файлу `mysql-connector-java-5.1.6-bin.jar`

Контрольные вопросы

Как должна выглядеть SQL-команда для получения данных о приземной температуре за весь январь 2009 года?

В каком каталоге должен находиться подключаемый файл драйвера MySQL?

Могут ли выполняться расчеты максимальной температуры при отсутствии подключения к сети Интернет?

Вопросы?

<http://www.chukin.ru/edu/java/>

chukin@rshu.ru

