MODUL

11

JDBC (JAVA DATABASE CONNECTIVITY)

Tujuan

Praktikan dapat memahami dan menjelaskan konsep polimorfisme dalam perograman serta dapat mengimplementasikan ke dalam pemrograman Java.

MENGENAL JDBC

Java Database Connectivity (JDBC) adalah API (driver) yang digunakan Java untuk melakukan koneksi dengan aplikasi lain atau dengan berbagai macam database. JDBC memungkinkan kita untuk membuat aplikasi Java yang dapat melakukan tiga hal, yaitu koneksi ke sumber data, mengirimkan query dan statement ke database, menerima dan mengolah resultset yang diperoleh dari database.

Database driver untuk setiap DBMS pada umumnya dapat didownload dari website pembuat DBMS tersebut. Beberapa vendor DBMS menyebut Database driver ini dengan sebutan Java Connector (J/Connector). Database driver biasanya dibungkus dalam fle yang berekstensi **jar**. Setiap database driver harus mengimplement interface java.sql.Driver.

Keunggulan JDBC:

- Mempertahankan data enterprise yang ada
- Menyederhanakan development enterprise
- Tidak memerlukan konfigurasi pada jaringan komputer.
- Akses penuh ke meta data
- Koneksi database menggunakan URL dan DataSource (yang menyediakan connection pooling dan distributed transaction).

JDBC API tersedia dalam paket java.sql dan javax.sql, didalamnya terdiri dari kelas-kelas antara lain:

- DriverManager: memanggil driver JDBC ke memori, dan dapat digunakan juga untuk membuka koneksi ke sumber data
- Connection: mempresentasikan suatu koneksi dengan suatu data source, juga digunakan untuk membuat objek Statement, PreparedStatement, dan CallableStatement
- Statement: mempresentasikan suatu perintah SQL, dan dapat digunakan untuk menerima objek ResultSet
- PreparedStatement: merupakan alternatif untuk objek Statement SQL yang telah terkompilasi awal
- CallableStatement: mempresentasikan suatu stored procedure, dan dapat digunakan untuk menjalankan stored procedures yang terkompilasi dalam suatu RDBMS yang mendukung fasilitas tersebut
- ResultSet: mempresentasikan sebuah hasil dari database yang dihasilkan dari statement SQL SELECT

- SQLException: suatu class exception yang membungkus kesalahan (error) pengaksesan database.

MELAKUKAN KONEKSI DATABASE

Untuk melakukan koneksi ke database pastikan komputer telah terinstal DBMS (database management system) yang akan digunakan. Contoh pada bab ini menggunakan DBMS MySQL yang dapat didownload di http://www.mysql.com. Berikut adalah langkah-langkah untuk melakukan koneksi ke database:

a. Mengaktifkan driver JDBC

```
Class.forName("com.mysql.jdbc.Driver");
```

b. Membuat koneksi

Langkah berikutnya adalah melakukan koneksi ke database, berikut perintahnya:

```
Conncection cn = DriverManager.getConnection(url, usr,
pwd);
```

dimana:

- cn adalah variabel bertipe Connection
- url adalah informasi jenis database, host database, nama database
- usr adalah user untuk koneksi ke database
- pwd adalah password yang digunakan untuk masuk ke database

Berikut adalah contoh kode program lengkap untuk melakukan koneksi:

Program 11-1

```
// ConnectDB.java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class connecDB {
    private String url, usr, pwd, db;
    connecDB() {
        db = "akademik";
        url = "jdbc:mysql://localhost/" + db;
        usr = "root";
        pwd = "";
    }
```

```
    Connection getConnect() {
        Connection cn = null;
        try {
            // Load driver database
            Class.forName("com.mysql.jdbc.Driver");
            cn = DriverManager.getConnection(url, usr,
            pwd); System.out.println("Koneksi Berhasil");
        } catch (ClassNotFoundException er ) {


        } catch (SQLException er) {
            System.out.println("Error #2: " +
            er.getMessage()); System.exit(0);
        }
        return cn;
    }
    public static void main(String [] args)
        { new connecDB().getConnect();
    }
}
```

Hasil Output

```
Koneksi Berhasil
```

MENAMPILKAN ISI DATABASE

Selanjutnya adalah memanipulasi database seperti melihat, menambahkan, mengubah, dan menghapus isi tabel atau biasa dikenal dengan istilah CRUD (create read update delete). Langkah awal yang harus disiapkan adalah database-nya terlebih dahulu. Sebagai contoh kita gunakan database Akademik pada tabel Mahasiswa. Berikut adalah query-nya:

```
-- Database: `akademik`
CREATE DATABASE `akademik`;
USE `akademik`;

-- Struktur dari tabel `mahasiswa` CREATE
TABLE IF NOT EXISTS `mahasiswa` (
  `NIM` varchar(9) NOT NULL, `Nama`
  varchar(50) DEFAULT NULL,
  `Alamat` varchar(50) DEFAULT
  NULL, PRIMARY KEY (`NIM`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
-- Dumping data untuk tabel `mahasiswa`
INSERT INTO `mahasiswa` (`NIM`, `Nama`, `Alamat`)
VALUES ('G0609001', 'Rendi', 'Metro'),
('G0609002', 'Jajat', 'Pringsewu'),
('G0609003', 'Santi', 'Kalianda'),
('G0609004', 'Budi', 'Bandar Lampung'),
('G0609005', 'Tuti', 'Bandar Jaya'),
('G0609006', 'Adang', 'Krui'),
('G0609007', 'Supri', 'Kotabumi'),
('G0609008', 'Kaka', 'Menggala'),
('G0609009', 'Ahmad', 'Talang Padang'),
('G0609010', 'Agus', 'Bandar Lampung');
```

Untuk menampung data yang diambil dari database kita memerlukan suatu class yang mengimplement interface **ResultSet**. Instance dari object bertipe ResultSet diperlukan untuk menampung hasil kembalian data dari database. Sebelum kita bisa memperoleh instance dari ResultSet, kita harus membuat instance dari class **Statement**. Class Statement mempunyai method executeQuery yang digunakan untuk menjalankan perintah query dalam database kemudian mengembalikan data hasil eksekusi query ke dalam object ResultSet.

Berikut ini adalah contoh kode untuk membuat instance class Statement, kemudian menjalankan query untuk mengambil data dari database yang hasilnya dipegang oleh ResultSet:

Program 11-2

```
// LihatMahasiswa.java import
java.sql.Connection; import
java.sql.Statement; import
java.sql.DriverManager; import
java.sql.ResultSet; import
java.sql.ResultSetMetaData;
import java.sql.SQLException;

 public class LihatMahasiswa {
    // launch the application
    public static void main( String args[] ) {
        Connection connection = null; // manages connection
        Statement statement = null; // query statement
        ResultSet resultSet = null; // manages results
        // JDBC driver name and database URL

        String DRIVER = "com.mysql.jdbc.Driver";
        String DB_URL = "jdbc:mysql://localhost/akademik";
        String usr = "root";
        String pwd = "";
        // connect to database books and query
        database try {
            // load driver class
            Class.forName(DRIVER);
            // establish connectionto database
            connection = DriverManager.getConnection(DB_URL,usr, pwd);
            // create statement for querying database
            statement = connection.createStatement();
```

```
            // query database
            String q = "SELECT * FROM Mahasiswa";
            resultSet = statement.executeQuery(q);
            // process query results
            ResultSetMetaData metaData = resultSet.getMetaData();
            int numColom = metaData.getColumnCount();
            System.out.println("Database Mahasiswa: ");

            for(int i=1; i<= numColom; i++)
                System.out.printf("%-8s\t", metaData.getColumnName(i));
            System.out.println();
            while(resultSet.next()) {
                for(int i=1; i<=numColom; i++)
                    System.out.printf("%-8s\t", resultSet.getObject(i));
                System.out.println();
            } // end while
        } catch (SQLException er) {
            er.printStackTrace();
        } catch (ClassNotFoundException er)
            { er.printStackTrace();
        } finally {
            try {
                resultSet.close();
                statement.close();
                connection.close();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    } // end main
}
```

Hasil Output

```
Database Mahasiswa:
NIM         Nama        Alamat
G0609001    Rendi       Metro
G0609002    Jajat       Pringsewu
G0609003    Santi       Kalianda
G0609004    Budi        Bandar Lampung
G0609005    Tuti        Bandar Jaya
G0609006    Adang       Krui
G0609007    Supri       Kotabumi
G0609008    Kaka        Menggala
G0609009    Ahmad       Talang Padang
G0609010    Agus        Bandar Lampung
```

Berikut ini adalah contoh kode program untuk memanipulasi database pada tabel Mahasiswa dengan menggunakan GUI Swing, terdapat tiga kelas yaitu Mahasiswa, ConnecDB, dan FormMahasiswa:

Program 11-3

```
// Mahasiswa.java
public class Mahasiswa {
    private String npm, nama, alamat;
```

```
    public String getNpm() {
        return npm;
    }
    public void setNpm(String npm) {
        this.npm = npm;
    }
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public String getAlamat() {
        return alamat;
    }
    public void setAlamat(String alamat)
        { this.alamat = alamat;
    }
}
// connecDB.java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class connecDB {
    private String url, usr, pwd, db;

    public connecDB() {
        db = "akademik";
        url = "jdbc:mysql://localhost/" + db;
        usr = "root";
        pwd = "";
    }
    public Connection getConnect() {
        Connection cn = null;
        try {
            // Load driver database
            Class.forName("com.mysql.jdbc.Driver");
            cn = DriverManager.getConnection(url, usr,
            pwd); //System.out.println("Koneksi Berhasil");
        } catch (ClassNotFoundException er) {
            System.out.println("Error #1: " +
            er.getMessage()); System.exit(0);
        } catch (SQLException er) {
```

```
            System.out.println("Error #2: " +
            er.getMessage()); System.exit(0);
        }
        return cn;
    }
}
// FormMahasiswa.java
import java.awt.event.*;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class FormMahasiswa extends JFrame {
    private String [] judul = {"NPM", "Nama",
    "Alamat"}; DefaultTableModel df;
    JTable tab = new JTable();
    JScrollPane scp = new JScrollPane();
    JPanel pnl = new JPanel();
    JLabel lblnama=new JLabel("Nama");
    JTextField txnama=new JTextField(20);
    JLabel lblnpm=new JLabel("NPM");
    JTextField txnpm=new JTextField(7);
    JLabel lblalamat = new JLabel("Alamat");
    JTextArea alamat = new JTextArea();
    JScrollPane sca = new JScrollPane(alamat);
    JButton btadd = new JButton("Simpan");
    JButton btnew = new JButton("Baru");
    JButton btdel = new JButton("Hapus");
    JButton btubh = new JButton("Ubah");

    FormMahasiswa() {
        super("Data Mahasiswa");
        setSize(460, 300);
        pnl.setLayout(null);
        pnl.add(lblnpm);
        lblnpm.setBounds(20, 10, 80, 20);
        pnl.add(txnpm);
        txnpm.setBounds(105,10,100,20);
        pnl.add(lblnama);
        lblnama.setBounds(20, 33, 80, 20);
        pnl.add(txnama);
        txnama.setBounds(105,33,175,20);
        pnl.add(lblalamat);
        lblalamat.setBounds(20, 56, 80, 20);
        pnl.add(sca);
        sca.setBounds(105, 56, 175, 45);
        pnl.add(btnew);
        btnew.setBounds(300, 10, 125, 20);
        btnew.addActionListener(new ActionListener()
            { @Override
            public void actionPerformed(ActionEvent e) {
```

```
                btnewAksi(e);
            }
        });
        pnl.add(btadd);
        btadd.setBounds(300, 33, 125, 20);
        btadd.addActionListener(new ActionListener()
            { @Override
            public void actionPerformed(ActionEvent e)
                { btaddAksi(e);
            }
        });
        pnl.add(btubh);
        btubh.setBounds(300, 56, 125, 20);
        btubh.setEnabled(false);
        btubh.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e)
                { btubhAksi(e);
            }
        });
        pnl.add(btdel);
        btdel.setBounds(300, 79, 125, 20);
        btdel.setEnabled(false);
        btdel.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e)
                { btdelAksi(e);
            }
        });
        df = new DefaultTableModel(null, judul);
        tab.setModel(df);
        scp.getViewport().add(tab);
        tab.setEnabled(true);
        tab.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent evt)
                { tabMouseClicked(evt);
            }
        });
        scp.setBounds(20, 110, 405, 130);
        pnl.add(scp);
        getContentPane().add(pnl);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    void loadData() {
        try{
            Connection cn = new
            connecDB().getConnect(); Statement st =
            cn.createStatement(); String sql = "SELECT
            * FROM Mahasiswa"; ResultSet rs =
            st.executeQuery(sql); while(rs.next()) {
```

```
                String npm, nama, alamat;
                npm = rs.getString("NIM");
                nama = rs.getString("Nama");
                alamat = rs.getString("Alamat");
                String [] data = {npm, nama, alamat};
                df.addRow(data);
            }
            rs.close();
            cn.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    void clearTable() {
        int numRow = df.getRowCount();
        for(int i=0; i<numRow; i++) {
            df.removeRow(0);
        }
    }

    void clearTextField() {
        txnpm.setText(null);
        txnama.setText(null);
        alamat.setText(null);
    }

    void simpanData(Mahasiswa M) {
        try{
            Connection cn = new connecDB().getConnect();
            Statement st = cn.createStatement();
            String sql = "INSERT INTO Mahasiswa (NIM, Nama, Alamat) "
                    + "VALUES ('" + M.getNpm() + "', '" + M.getNama()
                    + "', '" + M.getAlamat() +"')";
            int result = st.executeUpdate(sql);
            cn.close();
            JOptionPane.showMessageDialog(null, "Data berhasil disimpan",
                    "Info Proses", JOptionPane.INFORMATION_MESSAGE);
            String [] data = {M.getNpm(), M.getNama(),
            M.getAlamat()}; df.addRow(data);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    void hapusData(String npm) {
        try{
            Connection cn = new connecDB().getConnect();
            Statement st = cn.createStatement();
            String sql = "DELETE FROM Mahasiswa WHERE NIM =
            '"+npm+"'"; int result = st.executeUpdate(sql);
            cn.close();
```

```
            JOptionPane.showMessageDialog(null, "Data berhasil dihapus",
                    "Info Proses", JOptionPane.INFORMATION_MESSAGE);
            df.removeRow(tab.getSelectedRow());
            clearTextField();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    void ubahData(Mahasiswa M, String npm)
        { try{
            Connection cn = new connecDB().getConnect();
            Statement st = cn.createStatement();
            String sql = "UPDATE Mahasiswa SET NIM='" + M.getNpm()
                    + "', Nama ='" + M.getNama() + "', Alamat='" +
                    M.getAlamat() +"' WHERE NIM='" + npm + "'";
            int result = st.executeUpdate(sql);
            cn.close();
            JOptionPane.showMessageDialog(null, "Data berhasil diubah",
                    "Info Proses", JOptionPane.INFORMATION_MESSAGE);
            clearTable();
            loadData();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    private void btnewAksi(ActionEvent evt) {
        txnpm.setText(null);
        txnama.setText(null);
        alamat.setText(null);
        btubh.setEnabled(false);
        btdel.setEnabled(false);
        btadd.setEnabled(true);
    }

    private void btaddAksi(ActionEvent evt)
        { Mahasiswa M = new Mahasiswa();
        M.setNpm(txnpm.getText());
        M.setNama(txnama.getText());
        M.setAlamat(alamat.getText());
        simpanData(M);
    }

    private void btdelAksi(ActionEvent evt)
        { int status;
        status = JOptionPane.showConfirmDialog(null, "Yakin data
akan dihapus", "Konfirmasi", JOptionPane.OK_CANCEL_OPTION);
        if(status == 0) {
            hapusData(txnpm.getText());
        }
    }
```

```
    private void btubhAksi(ActionEvent evt)
        { Mahasiswa M = new Mahasiswa();
        M.setNpm(txnpm.getText());
        M.setNama(txnama.getText());
        M.setAlamat(alamat.getText());
        ubahData(M, tab.getValueAt(tab.getSelectedRow(), 0).toString());
    }

    private void tabMouseClicked(MouseEvent evt) {
        btubh.setEnabled(true);
        btdel.setEnabled(true);
        btadd.setEnabled(false);
        String npm, nama, alt;
        npm = tab.getValueAt(tab.getSelectedRow(), 0).toString();
        nama = tab.getValueAt(tab.getSelectedRow(), 1).toString();
        alt = tab.getValueAt(tab.getSelectedRow(), 2).toString();
        txnpm.setText(npm);
        txnama.setText(nama);
        alamat.setText(alt);
    }

    public static void main(String [] args)
        { new FormMahasiswa().loadData();
    }
}
```