**LEMBAR JAWABAN UAS**

**PEMROGRAMAN BERORIENTASI OBYEK (JAVA II)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama | : | Arif Frima Ari Suwadji | Mata Kuliah | : | PEMROGRAMAN BERORIENTASI OBYEK (JAVA II) |
| NIM | : | 221011700443 | Nama Dosen | : | HANANING SUMANINGDIAH LARASATI |
| Semester / Kelas | : | 04 / 04SIFE003 |  |  |

**Jawaban!**

1. **Import statements:**

* import java.sql.Connection;

Connection adalah interface utama dalam JDBC untuk mengelola koneksi ke database. Ini menyediakan metode untuk membuat Statement, PreparedStatement, dan CallableStatement, serta mengatur transaksi database.

* import java.sql.DriverManager;

DriverManager adalah kelas yang mengelola set driver JDBC. Fungsi utamanya adalah untuk membuat koneksi ke database menggunakan URL JDBC yang sesuai.

* import java.sql.SQLException;

SQLException adalah exception yang dilempar ketika terjadi kesalahan dalam mengakses database. Ini mencakup berbagai kesalahan seperti kegagalan koneksi, kesalahan sintaks SQL, atau masalah izin.

* import java.sql.Statement;

Statement adalah interface untuk mengeksekusi perintah SQL statis. Ini digunakan untuk mengirim perintah SQL ke database dan mengambil hasilnya.

* import java.sql.ResultSet;

ResultSet merepresentasikan hasil dari query database. Ini menyediakan metode untuk mengakses data yang diambil oleh query, seperti getString(), getInt(), dll.

* import javax.swing.JOptionPane;

JOptionPane adalah Class dalam Swing GUI toolkit yang digunakan untuk menampilkan dialog box standar. Ini sering digunakan untuk menampilkan pesan, konfirmasi, atau input dari pengguna dalam aplikasi GUI.

1. **Connection con = DriverManager.getConnection(host, username, password);**

Kode ini adalah langkah krusial dalam membuat koneksi ke database menggunakan JDBC:

* Connection: Ini adalah interface yang merepresentasikan sesi koneksi dengan database spesifik. Setiap perintah SQL dieksekusi dan hasilnya dikembalikan dalam konteks koneksi ini.
* DriverManager.getConnection(): Ini adalah metode statis yang mencoba membuat koneksi ke database. Metode ini melakukan beberapa hal:
  1. Mencari driver JDBC yang sesuai berdasarkan URL yang diberikan.
  2. Membuat instance dari driver tersebut.
  3. Menggunakan driver untuk membuat koneksi ke database.
* Parameter:
* host: URL JDBC yang menentukan lokasi dan jenis database. Format umumnya adalah "jdbc:subprotocol:subname". Contoh:
  + MySQL: "jdbc:mysql://localhost:3306/nama\_database"
  + PostgreSQL: "jdbc:postgresql://localhost:5432/nama\_database"
  + Oracle: "jdbc:oracle:thin:@localhost:1521:nama\_database"
* username: Nama pengguna yang memiliki akses ke database.
* password: Kata sandi yang sesuai dengan nama pengguna.

1. **Query SQL dan Data Type:**

String SQL = "SELECT \* FROM Workers"; String id = Integer.toString(id\_col);

a. String SQL = "SELECT \* FROM Workers";

* String SQL adalah deklarasi string yang berisi query SQL.
* SELECT \* : Ini adalah perintah SQL untuk memilih semua kolom dari tabel yang ditentukan.
* FROM Workers : Menentukan tabel 'Workers' sebagai sumber data.
* Query ini akan mengambil semua baris dan kolom dari tabel Workers.
* Penggunaan \* bisa tidak efisien untuk tabel besar, lebih baik menyebutkan kolom spesifik yang dibutuhkan.

b. String id = Integer.toString(id\_col);

* String id adalah operasi konversi tipe data dari Integer ke String.
* id\_col diasumsikan sebagai variabel bertipe Integer.
* Integer.toString() adalah metode statis yang mengkonversi nilai Integer menjadi representasi String-nya.

1. **Potongan kode actionPerformed():**

public void actionPerformed(ActionEvent e) {

try {

String kode = txtKode.getText();

String SQL = "DELETE FROM tblbarang WHERE kode='"+kode+"'";

Statement stmt = con.createStatement();

int i = stmt.executeUpdate(SQL);

if (i == 1) {

JOptionPane.showMessageDialog(null, "Data berhasil dihapus");

}

Bersih();

TampilData();

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, ex.getMessage());

}

}

1. Public void actionPerformed(ActionEvent e):
   * Ini adalah metode yang dipanggil ketika suatu aksi terjadi (misalnya, tombol ditekan).
   * ActionEvent e adalah objek yang berisi informasi tentang event yang terjadi.
2. String kode = txtKode.getText();
   * Mengambil teks dari komponen GUI (mungkin JTextField) yang bernama txtKode.
   * Nilai ini akan digunakan sebagai kriteria untuk menghapus data.
3. String SQL = "DELETE FROM tblbarang WHERE kode='"+kode+"'";
   * Membuat query SQL untuk menghapus data dari tabel tblbarang.
   * Perhatikan: Cara ini rentan terhadap SQL injection. Lebih aman menggunakan PreparedStatement.
4. Statement stmt = con.createStatement();
   * Membuat objek Statement untuk mengeksekusi SQL.
   * con diasumsikan sebagai objek Connection yang sudah dibuat sebelumnya.
5. int i = stmt.executeUpdate(SQL);
   * Mengeksekusi query DELETE.
   * Mengembalikan jumlah baris yang terpengaruh oleh operasi.
6. if (i == 1) { ... }
   * Memeriksa apakah satu baris telah dihapus.
   * Jika ya, menampilkan pesan sukses menggunakan JOptionPane.
7. Bersih(); dan TampilData();
   * Metode kustom yang mungkin membersihkan form input dan memperbarui tampilan data.
8. Penanganan Exception => catch (Exception ex):
   * Menggunakan try-catch untuk menangani Exception.
   * Jika terjadi error, pesan error ditampilkan menggunakan JOptionPane.
9. **Parameter ResultSet:**

ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE:

1. ResultSet.TYPE\_SCROLL\_INSENSITIVE:
   * Tipe ini memungkinkan cursor ResultSet untuk bergerak maju dan mundur.
   * "Insensitive" berarti ResultSet tidak akan mencerminkan perubahan yang dilakukan oleh transaksi lain pada database selama ResultSet masih terbuka.
   * Keuntungan: Fleksibilitas dalam navigasi data.
   * Kerugian: Memori tambahan diperlukan karena seluruh hasil query disimpan di memori client.
2. ResultSet.CONCUR\_UPDATABLE:
   * Memungkinkan ResultSet untuk diupdate.
   * Anda dapat memodifikasi data dalam ResultSet dan menyimpan perubahan kembali ke database menggunakan metode seperti updateString(), updateInt(), dll., diikuti dengan updateRow() atau insertRow().
   * Keuntungan: Memungkinkan modifikasi data langsung melalui ResultSet.
   * Kerugian: Performa mungkin lebih lambat dibandingkan dengan ResultSet read-only.