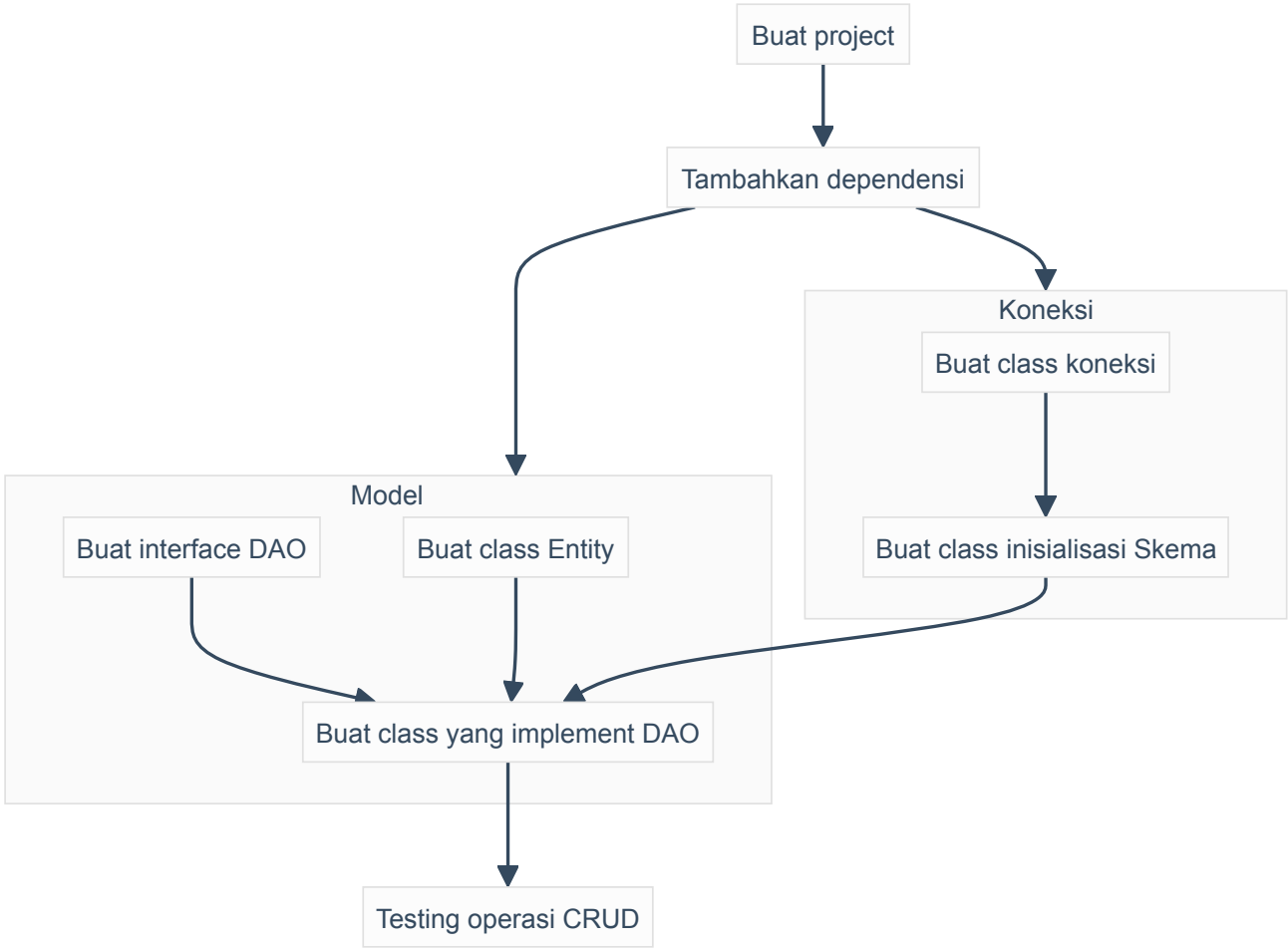
PBO - Konsep Akses SQLite DB Berbasis DAO

Alur Koneksi Basis Data

Langkah pembuatan program Java yang terkoneksi dengan basis data SQLite sebagaimana ditampilkan pada diagram di bawah ini.



Catatan Tambahan:

- Pada class koneksi kita akan mengimplementasikan Singleton Pattern dengan tujuan:
  - Efisiensi sumber daya
  - Kontrol terpusat
  - Konsistensi koneksi
  - Kemudahan pengelolaan transaksi
- Pattern yang digunakan pada akses data adalah DAO dengan tujuan:
  - Pemilahan kode (*separation of concern*)
  - Peningkatan readability dan maintainability
  - Portabilitas dan fleksibilitas

Langkah Implementasi Pengerjaan

1. Setup proyek menggunakan gradle, jangan lupa menambahkan dependensi SQLite JDBC driver berikut pada file `build.gradle`:

```
dependencies {
        // ...
        implementation 'org.xerial:sqlite-jdbc:3.36.0.3'
        // ...
}
```

2. Buat class `DBConnect` untuk koneksi basis data: pada tahap ini kita sebaiknya membuat class yang mengikuti patter singleton untuk koneksi ke basis data.

```
public class DBConnect {
    private static final String URL = "jdbc:sqlite:mahasiswa.db";
    private static Connection instance;

    private DBConnect() {
        // Private constructor to prevent instantiation
    }

    public static Connection connect() throws SQLException {
        if (instance == null || instance.isClosed()) {
            try {
                instance = DriverManager.getConnection(URL);
                // System.out.println("Koneksi ke SQLite berhasil.");
            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
        return instance;
    }
}
```

3. Buat class `DBSetup` untuk menyiapkan skema tabel, semacam skrip migrasi yang kita buat secara manual. Class ini nanti akan dipanggil setiap aplikasi dijalankan dan bertugas untuk mengecek apakah skema basis data sudah siap. Jika belum siap, class ini juga bertanggung jawab untuk mengeksekusi kode pembuatan basis data.

```
import java.sql.Connection;
import java.sql.Statement;
import java.sql.SQLException;

public class DBSetup {
    public static void createTable() {
        String sql = "CREATE TABLE IF NOT EXISTS mahasiswa (\n"
                   + " nim TEXT PRIMARY KEY,\n"
                   + " nama TEXT NOT NULL,\n"
                   + " ipk REAL,\n"
                   + " total_sks INTEGER\n"
                   + ");";

        try (Connection conn = DBConnect.connect();
             Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
            // System.out.println("Tabel mahasiswa telah dibuat atau sudah ada.");
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
```

```
    }
}
```

4. Buat model entitas `Mahasiswa` yang merupakan mapping skema basis data ke dalam object. Jangan lupa tambahkan getter, setter, serta beberapa logika yang diperlukan.

```
public class Mahasiswa {
    private String nim;
    private String nama;
    private double ipk;
    private int totalSks;

    // Constructor, getters, and setters

    public Mahasiswa(String nim, String nama, double ipk, int totalSks) {
        this.nim = nim;
        this.nama = nama;
        this.ipk = ipk;
        this.totalSks = totalSks;
    }

    public String getNim() {
        return nim;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public double getIpk() {
        return ipk;
    }

    public void setIpk(double ipk) {
        this.ipk = ipk;
    }

    public int getTotalSks() {
        return totalSks;
    }

    public void setTotalSks(int totalSks) {
        this.totalSks = totalSks;
    }

        @Override
    public String toString() {
        return "["+nim+"] "+nama+" : "+ipk+" ("+totalSks+" SKS)";
    }
}
```

5. Buat DAO interface untuk masing-masing model entitas. Di sini kita juga mendefinisikan method operasi apa saja (minimal CRUD, tapi juga bisa ditambah yang lain) yang dapat dilakukan oleh entitas tersebut.

```
import java.util.List;

public interface MahasiswaDAO {
    void insert(Mahasiswa mahasiswa);
    Mahasiswa selectByNim(String nim);
    List<Mahasiswa> selectAll();
    void update(Mahasiswa mahasiswa);
    void delete(String nim);
}
```

5. Buat class untuk mengimplementasikan DAO. Di sini kita mengimplementasikan koding CRUD dan operasi basis data lain sesuai dengan jenis basis data yang digunakan.

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class MahasiswaDAOSQLite implements MahasiswaDAO {

    @Override
    public void insert(Mahasiswa mahasiswa) {
        String sql = "INSERT INTO mahasiswa(nim, nama, ipk, total_sks) VALUES(?,?,?,?)";

        try (Connection conn = DBConnect.connect();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, mahasiswa.getNim());
            pstmt.setString(2, mahasiswa.getNama());
            pstmt.setDouble(3, mahasiswa.getIpk());
            pstmt.setInt(4, mahasiswa.getTotalSks());
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public Mahasiswa selectByNim(String nim) {
        String sql = "SELECT nim, nama, ipk, total_sks FROM mahasiswa WHERE nim = ?";
        Mahasiswa mahasiswa = null;

        try (Connection conn = DBConnect.connect();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nim);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                mahasiswa = new Mahasiswa(
                        rs.getString("nim"),
                        rs.getString("nama"),
                        rs.getDouble("ipk"),
                        rs.getInt("total_sks")
                );
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
```

```
        }

        return mahasiswa;
    }

    @Override
    public List<Mahasiswa> selectAll() {
        String sql = "SELECT nim, nama, ipk, total_sks FROM mahasiswa";
        List<Mahasiswa> mahasiswas = new ArrayList<>();

        try (Connection conn = DBConnect.connect();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                Mahasiswa mahasiswa = new Mahasiswa(
                        rs.getString("nim"),
                        rs.getString("nama"),
                        rs.getDouble("ipk"),
                        rs.getInt("total_sks")
                );
                mahasiswas.add(mahasiswa);
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return mahasiswas;
    }

    @Override
    public void update(Mahasiswa mahasiswa) {
        String sql = "UPDATE mahasiswa SET nama = ?, ipk = ?, total_sks = ? WHERE nim =
?";

        try (Connection conn = DBConnect.connect();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, mahasiswa.getNama());
            pstmt.setDouble(2, mahasiswa.getIpk());
            pstmt.setInt(3, mahasiswa.getTotalSks());
            pstmt.setString(4, mahasiswa.getNim());
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public void delete(String nim) {
        String sql = "DELETE FROM mahasiswa WHERE nim = ?";

        try (Connection conn = DBConnect.connect();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nim);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

7. Uji coba serta menjalankannya pada fungsi main.

```
public class Main {
    public static void main(String[] args) {
        DBSetup.createTable();

        MahasiswaDAO mahasiswaDAO = new MahasiswaDAOSQLite();

        // Insert Data
        mahasiswaDAO.insert(new Mahasiswa("A123456", "Budi", 3.5, 140));
        mahasiswaDAO.insert(new Mahasiswa("A234567", "Leni", 3.7, 120));
        mahasiswaDAO.insert(new Mahasiswa("B123456", "Hasan", 3.2, 135));
        mahasiswaDAO.insert(new Mahasiswa("B234567", "Ferdinan", 3.8, 112));
        mahasiswaDAO.insert(new Mahasiswa("C123456", "Roni", 3.1, 110));

        // Select Data
        Mahasiswa selectedMhs = mahasiswaDAO.selectByNim("B234567");
        System.out.println(selectedMhs);

        // Update Data
        selectedMhs.setNama("Frida");
        mahasiswaDAO.update(selectedMhs);

        List<Mahasiswa> dataMhs = mahasiswaDAO.selectAll();
        dataMhs.forEach(mahasiswa -> System.out.println(mahasiswa));

        System.out.println("====================================");

        // Delete Data
        mahasiswaDAO.delete("A123456");
        dataMhs = mahasiswaDAO.selectAll();
        dataMhs.forEach(mahasiswa -> System.out.println(mahasiswa));
    }
}
```

Konsep Lanjutan

Ada dua hal yang dibahas pada bagian ini:

1. Dependency injection untuk meningkatkan efektifikasi koneksi basis data
2. Penambahan studi kasus penggunaan relasi tabel Mahasiswa dengan Dosen

Dependency Injection

Contoh koneksi dan query basis data sebelum kita selalu membuka dan menutup koneksi basis data pada setiap eksekusi query, di mana tujuannya adalah untuk mencegah kebocoran koneksi. Meski demikian, cara tersebut akan membuat aplikasi berjalan lebih lambat karena harus selalu membuka dan menutup koneksi pada setiap ekskusi query.

Alternatif solusinya dengan menggunakan Dependency Injection, di mana class implementasi DAO menggunakan koneksi yang diberikan melalui constructor. Dengan demikian, sebagai programmer kita bisa lebih fleksibel menetukan kapan koneksi akan dibuka dan ditutup pada konteks aktivitas / fungsi yang memerlukan banyak query basis data.

Berikut adalah langkahnya:

- Buat class baru `MahasiswaModel` dengan melakukan copy paste dari class `MahasiswaDAOSQLite` serta melakukan beberapa perubahan sebagai berikut:

```
public class MahasiswaModel implements MahasiswaDAO {

    private Connection conn;

    public MahasiswaModel(Connection conn) {
        this.conn = conn;
    }

    @Override
    public void insert(Mahasiswa mahasiswa) {
        // ...

        try (PreparedStatement pstmt = conn.prepareStatement(sql)){
        // ...
        }
    }

    @Override
    public Mahasiswa selectByNim(String nim) {
        // ...

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            // ...
        }

        return mahasiswa;
    }

    @Override
    public List<Mahasiswa> selectAll() {
        // ...

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            // ...
        }

        return mahasiswas;
    }

    @Override
    public void update(Mahasiswa mahasiswa) {
        // ...

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            // ...
        }
    }

    @Override
    public void delete(String nim) {
        // ...

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            // ...
        }
```

```
    }
}
```

- Selanjutnya untuk pemanggilannya pada class main, kita merubahnya menjadi seperti di bawah ini:

```
public static void main(String[] args) {
        DBSetup.createTable();

        try (Connection conn = DBConnect.connect()) {
                MahasiswaDAO mahasiswaDAO = new MahasiswaModel(conn);

                // ...
        } catch (SQLException e) {
                e.printStackTrace();
        }
}
```

Relasi Tabel

- Ubah query pada class `DBSetup` menjadi seperti di bawah ini:

```
public class DBSetup {
    public static void createTable() {
        String [] sql = {
                """
                    CREATE TABLE IF NOT EXISTS dosen (
                    npp TEXT PRIMARY KEY,
                    nama TEXT NOT NULL);""",

                                """
                                        CREATE TABLE IF NOT EXISTS mahasiswa (
                                        nim TEXT PRIMARY KEY,
                                        nama TEXT NOT NULL,
                                        ipk REAL,
                                        total_sks INTEGER,
                                        npp_dosen TEXT,
                                        FOREIGN KEY (npp_dosen) REFERENCES dosen(npp));"""

            };

        try (Connection conn = DBConnect.connect();
             Statement stmt = conn.createStatement()) {
                for(String query : sql) stmt.execute(query);
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Buat class Entity `Dosen` dengan isian sederhana sebagai berikut:

```
public class Dosen {
    public String npp;
    public String nama;
    public List<Mahasiswa> listPerwalian;
```

```
    public Dosen(String npp, String nama) {
        this.npp = npp;
        this.nama = nama;
        this.listPerwalian = new ArrayList<>();
    }

    public Dosen(String npp, String nama, List<Mahasiswa> listPerwalian) {
        this.npp = npp;
        this.nama = nama;
        this.listPerwalian = listPerwalian;
    }
}
```

- Buat interface `DosenDAO`

```
public interface DosenDAO {
    void insert(Dosen dosen);
    Dosen selectByNpp(String npp);
    List<Dosen> selectAll();
    void update(Dosen dosen);
    void delete(String npp);
    List<Mahasiswa> getMahasiswaPerwalian(Dosen dosen);
}
```

- Buat class implementasi `DosenModel` dengan isian sebagai berikut:

```
public class DosenModel implements DosenDAO {

    Connection conn;

    public DosenModel(Connection conn){
        this.conn = conn;
    }

    @Override
    public void insert(Dosen dosen) {
        String sql = "INSERT INTO dosen(npp, nama) VALUES(?, ?)";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, dosen.npp);
            pstmt.setString(2, dosen.nama);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public Dosen selectByNpp(String npp) {
        String sql = "SELECT npp, nama FROM dosen WHERE npp = ?";
        Dosen dosen = null;

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, npp);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                dosen = new Dosen(
```

```
                        rs.getString("npp"),
                        rs.getString("nama"));
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return dosen;
    }

    @Override
    public List<Dosen> selectAll() {
        String sql = "SELECT npp, nama FROM dosen";
        List<Dosen> dosens = new ArrayList<>();

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                Dosen dosen = new Dosen(
                        rs.getString("npp"),
                        rs.getString("nama"));
                dosens.add(dosen);
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return dosens;
    }

    @Override
    public void update(Dosen dosen) {
        String sql = "UPDATE dosen SET nama = ? WHERE npp = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, dosen.nama);
            pstmt.setString(2, dosen.npp);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public void delete(String npp) {
        String sql = "DELETE FROM dosen WHERE npp = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, npp);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public List<Mahasiswa> getMahasiswaPerwalian(Dosen dosen) {
        String sql = "SELECT m.nim, m.nama, m.ipk, m.total_sks, d.npp, d.nama AS
dosen_nama "
                   + "FROM mahasiswa m "
```

```
                    + "LEFT JOIN dosen d ON m.dosen_wali_npp = d.npp "
                    + "WHERE d.npp = ?";
        List<Mahasiswa> mahasiswaPerwalian = new ArrayList<>();

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, dosen.npp);
            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                Mahasiswa mahasiswa = new Mahasiswa(
                        rs.getString("nim"),
                        rs.getString("nama"),
                        rs.getDouble("ipk"),
                        rs.getInt("total_sks"));
                mahasiswaPerwalian.add(mahasiswa);
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return mahasiswaPerwalian;
    }
}
```

- Tambahkan atribut `dosenWali` pada class entity `Mahasiswa` sebagai berikut:

```
public class Mahasiswa {
    private String nim;
    private String nama;
    private double ipk;
    private int totalSks;
    public Dosen dosenWali;

        public Mahasiswa(String nim, String nama, double ipk, int totalSks) {
        this.nim = nim;
        this.nama = nama;
        this.ipk = ipk;
        this.totalSks = totalSks;
        this.dosenWali = null;
    }

        // ...

        @Override
    public String toString() {
        if (this.dosenWali == null)
            return "["+nim+"] "+nama+" : "+ipk+" ("+totalSks+" SKS)";
        else
            return "["+nim+"] "+nama+" : "+ipk+" ("+totalSks+" SKS) - Wali:
"+dosenWali.nama;
    }
}
```

- Ubah juga eksekusi query pada class `MahasiswaModel` menjadi seperti berikut:

```
public class MahasiswaModel implements MahasiswaDAO {

    private Connection conn;
```

```
    public MahasiswaModel(Connection conn) {
        this.conn = conn;
    }

    @Override
    public void insert(Mahasiswa mahasiswa) {
        String sql = "INSERT INTO mahasiswa(nim, nama, ipk, total_sks) VALUES(?,?,?,?)";
        if(mahasiswa.dosenWali != null){
            sql = "INSERT INTO mahasiswa(nim, nama, ipk, total_sks, npp_dosen)
VALUES(?,?,?,?,?)";
        }

        try (PreparedStatement pstmt = conn.prepareStatement(sql)){
            pstmt.setString(1, mahasiswa.getNim());
            pstmt.setString(2, mahasiswa.getNama());
            pstmt.setDouble(3, mahasiswa.getIpk());
            pstmt.setInt(4, mahasiswa.getTotalSks());
            if(mahasiswa.dosenWali != null){
                pstmt.setString(5, mahasiswa.dosenWali.npp);
            }
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Salahnya adalah: "+e.getMessage());
        }
    }

    @Override
    public Mahasiswa selectByNim(String nim) {
        String sql = """
                SELECT nim, mahasiswa.nama as nama, ipk, total_sks,
                        dosen.npp, dosen.nama as nama_wali
                    FROM mahasiswa
                    left join dosen on (dosen.npp = mahasiswa.npp_dosen)
                WHERE nim = ?""";
        Mahasiswa mahasiswa = null;

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nim);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                mahasiswa = new Mahasiswa(
                        rs.getString("nim"),
                        rs.getString("nama"),
                        rs.getDouble("ipk"),
                        rs.getInt("total_sks")
                );
                String npp_wali = rs.getString("npp");
                String nama_wali = rs.getString("nama_wali");
                if(npp_wali != null){
                    mahasiswa.dosenWali = new Dosen(npp_wali, nama_wali);
                }
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return mahasiswa;
    }
```

```
    @Override
    public List<Mahasiswa> selectAll() {

        String sql = """
            SELECT nim, mahasiswa.nama as nama, ipk, total_sks,
                    dosen.npp, dosen.nama as nama_wali
                FROM mahasiswa
                left join dosen on (dosen.npp = mahasiswa.npp_dosen)""";
        List<Mahasiswa> mahasiswas = new ArrayList<>();

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                Mahasiswa mahasiswa = new Mahasiswa(
                        rs.getString("nim"),
                        rs.getString("nama"),
                        rs.getDouble("ipk"),
                        rs.getInt("total_sks")
                );
                String npp_wali = rs.getString("npp");
                String nama_wali = rs.getString("nama_wali");
                if(npp_wali != null){
                    mahasiswa.dosenWali = new Dosen(npp_wali, nama_wali);
                }
                mahasiswas.add(mahasiswa);
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return mahasiswas;
    }

    @Override
    public void update(Mahasiswa mahasiswa) {
        String sql = "UPDATE mahasiswa SET nama = ?, ipk = ?, total_sks = ? ";
        if(mahasiswa.dosenWali != null){
            sql += ", npp_dosen = ? ";
        }
        sql += "WHERE nim = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, mahasiswa.getNama());
            pstmt.setDouble(2, mahasiswa.getIpk());
            pstmt.setInt(3, mahasiswa.getTotalSks());
            pstmt.setString(4, mahasiswa.getNim());
            if(mahasiswa.dosenWali != null){
                pstmt.setString(4, mahasiswa.dosenWali.npp);
                pstmt.setString(5, mahasiswa.getNim());
            } else {
                pstmt.setString(4, mahasiswa.getNim());
            }
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public void delete(String nim) {
```

```
        String sql = "DELETE FROM mahasiswa WHERE nim = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nim);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}
```