LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: RPL 2
Kelas	: 4IA06
Praktikum ke-	: 4
Tanggal	: 05 November 2024
Materi	: ORM dan Framework Hibernate
NPM	: 50421535
Nama	: Fizri Rosdiansyah
Ketua Asisten	: Gilbert Jefferson Faozato Mendrofa
Paraf Asisten	:
Nama Asisten	:
Jumlah Lembar	: 13 Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2024

Rekayasa Perangkat Lunak 2

Soal

1. Jelaskan satu per satu Codingan kalian dari hasil screenshot Activity

Jawab

1. A. mahasiswaController.java

```
package com.mahasiswa.controller;

import com.mahasiswa.model.hibernateUtil;
import com.mahasiswa.model.modelMahasiswa;
import java.util.List;
import org.hibernate.*;
import org.hibernate.query.Query;

public class mahasiswaController {
    public void addMhs(modelMahasiswa mhs){
        Transaction trx = null;

        try(Session session = hibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            session.save(mhs);
            trx.commit();
        }catch (Exception e){
            if (trx != null){
                trx.rollback();
            }
            e.printStackTrace();
        }
    }

    public void updateMhs(modelMahasiswa mhs){
        Transaction trx = null;

        try (Session session = hibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            session.update(mhs);
            trx.commit();
        }catch (Exception e){
            if (trx != null){
                trx.rollback();
            }
            e.printStackTrace();
        }
    }

    public void deleteMhs(int id){
        Transaction trx = null;

        try (Session session = hibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            modelMahasiswa mhs = session.get(modelMahasiswa.class, id);
```

```
            if(mhs != null){
                session.delete(mhs);
                System.out.println("Berhasil dihapus");
            }
            trx.commit();
        }catch (Exception e){
            if (trx != null){
                trx.rollback();
            }
            e.printStackTrace();
        }
    }

    public List<modelMahasiswa> getAllMahasiswa(){
        Transaction trx = null;
        List<modelMahasiswa> listMhs = null;

        try (Session session = hibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            Query<modelMahasiswa> query = session.createQuery("from modelMahasiswa",
modelMahasiswa.class);
            listMhs = query.list();
            trx.commit();
        }catch (Exception e){
            if (trx != null){
                trx.rollback();
            }
            e.printStackTrace();
        }
        return listMhs;
    }
}
```

Kode di atas adalah sebuah controller dalam aplikasi Java yang menggunakan Hibernate untuk mengelola operasi CRUD (Create, Read, Update, Delete) pada entitas mahasiswa. Berikut adalah fungsi dari setiap metode dalam kelas mahasiswaController:

- **addMhs(modelMahasiswa mhs)**
  Fungsi ini menambahkan objek mahasiswa baru ke dalam database. Objek mahasiswa yang diterima sebagai parameter akan disimpan melalui session Hibernate. Jika terjadi kesalahan, transaksi akan di-rollback untuk membatalkan perubahan.
- **updateMhs(modelMahasiswa mhs)**
  Fungsi ini memperbarui data mahasiswa yang sudah ada di database. Menggunakan Hibernate untuk meng-update informasi dari objek modelMahasiswa yang diterima sebagai parameter.
- **deleteMhs(int id)**
  Fungsi ini menghapus data mahasiswa dari database berdasarkan ID yang diberikan sebagai parameter. Jika objek mahasiswa dengan ID tersebut ditemukan, maka objek tersebut akan dihapus dari database.
- **getAllMahasiswa()**
  Fungsi ini mengambil semua data mahasiswa dari database dalam bentuk List<modelMahasiswa>. Menggunakan query Hibernate untuk mengambil semua entitas modelMahasiswa yang ada dalam database.

B.hibernateUtil.java

```
package com.mahasiswa.model;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class hibernateUtil {
    private static SessionFactory sessionFactory;

    static{
        try{
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex){
            System.err.println("Initial SessionFactory creation failed" + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory(){
        return sessionFactory;
    }

    public static void testConnection(){
        try (Session session = sessionFactory.openSession()){
            System.out.println("Connection to the database was successfull");
        } catch (Exception e){
            System.err.println("Failed to connect to the database");
            e.printStackTrace();
        }
    }
}
```

Kode di atas adalah kelas hibernateUtil, yang menyediakan konfigurasi dan utilitas untuk mengelola koneksi database menggunakan Hibernate. Berikut penjelasan fungsi dan struktur dalam class ini:

- **SessionFactory sessionFactory**
  SessionFactory adalah objek Hibernate yang didefinisikan sebagai static, dan digunakan untuk membuat Session Hibernate untuk berinteraksi dengan database. Objek ini dibuat hanya sekali dan digunakan sepanjang aplikasi berjalan untuk menghindari overhead pembuatan koneksi berulang kali.
- **Static Block (static{...})**
  Blok static ini dijalankan sekali ketika kelas pertama kali dimuat. Di dalamnya, sessionFactory dikonfigurasi menggunakan file konfigurasi Hibernate (biasanya hibernate.cfg.xml). Jika terjadi kegagalan dalam konfigurasi, blok ini akan melempar ExceptionInInitializerError.
- **getSessionFactory()**
  Metode ini mengembalikan instance SessionFactory yang sudah dikonfigurasi. Metode ini memungkinkan bagian lain dalam aplikasi untuk mendapatkan Session yang akan digunakan dalam operasi CRUD.
- **testConnection()**
  Metode ini mencoba untuk membuka sesi dengan SessionFactory, lalu menutupnya, dan menampilkan pesan bahwa koneksi berhasil. Jika terjadi kesalahan, pesan error akan ditampilkan. Metode ini berguna untuk menguji apakah koneksi ke database berhasil.

C.modelMahasiswa.java

```
package com.mahasiswa.model;

import jakarta.persistence.*;

@Entity
@Table(name = "mahasiswa")
public class modelMahasiswa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "npm", nullable = false, length = 8)
    private String npm;

    @Column(name = "nama", nullable = false, length = 50)
    private String nama;

    @Column(name = "semester")
    private int semester;

    @Column(name = "ipk")
    private float ipk;

    public modelMahasiswa(){

    }

    public modelMahasiswa(int id, String npm, String nama, int semester, float ipk){
        this.id = id;
        this.npm = npm;
        this.nama = nama;
        this.semester = semester;
        this.ipk = ipk;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    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 int getSemester() {
        return semester;
    }

    public void setSemester(int semester) {
        this.semester = semester;
    }

    public float getIpk() {
        return ipk;
    }

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

Kode di atas adalah kelas modelMahasiswa, yang merupakan representasi dari tabel mahasiswa dalam database menggunakan Java Persistence API (JPA) dengan Hibernate. Berikut penjelasannya:

- **@Entity**
  Anotasi ini menandakan bahwa kelas modelMahasiswa adalah entitas JPA dan akan dipetakan ke tabel dalam database.
- **@Table(name = "mahasiswa")**
  Anotasi ini menentukan bahwa entitas modelMahasiswa akan dipetakan ke tabel bernama mahasiswa dalam database.
- **Atribut dan Anotasi Kolom**
- **@Id**
  Menandakan bahwa atribut id adalah primary key dalam tabel.
- **@GeneratedValue(strategy = GenerationType.IDENTITY)**
  Menentukan bahwa id adalah auto-increment, yang artinya nilainya akan diisi secara otomatis oleh database.
- **@Column(name = "column_name")**
  Anotasi ini menentukan nama kolom dalam tabel dan properti-propertinya seperti nullable dan length.
- **Atribut Lainnya (npm, nama, semester, ipk)**
  Masing-masing atribut ini memiliki anotasi @Column untuk menghubungkan properti kelas dengan kolom dalam tabel mahasiswa.
- **Konstruktor**
- **Default Constructor (modelMahasiswa())**
  Dibutuhkan oleh Hibernate untuk membuat instance tanpa menginisialisasi atribut.

- **Parameter Constructor**
  Konstruktor dengan parameter untuk mempermudah inisialisasi objek modelMahasiswa dengan nilai yang diberikan.
- **Getter dan Setter**
  Fungsi-fungsi ini memungkinkan pengaksesan dan pengubahan nilai atribut modelMahasiswa secara aman dari luar kelas.

D.modelTabelMahasiswa.java

```
package com.mahasiswa.model;

import java.util.List;
import javax.swing.table.AbstractTableModel;

public class modelTableMahasiswa extends AbstractTableModel{
   private List<modelMahasiswa> mahasiswaList;
   private String[] columnNames = {"ID", "NPM", "NAMA", "SEMESTER", "IPK"};

   public modelTableMahasiswa(List<modelMahasiswa> mahasiswaList){
      this.mahasiswaList = mahasiswaList;
   }

   @Override
   public int getRowCount(){
      return mahasiswaList.size();
   }

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

   @Override
   public Object getValueAt(int rowIndex, int columnIndex){
      modelMahasiswa mahasiswa = mahasiswaList.get(rowIndex);
      switch(columnIndex){
         case 0:
            return mahasiswa.getId();
         case 1:
            return mahasiswa.getNpm();
         case 2:
            return mahasiswa.getNama();
         case 3:
            return mahasiswa.getSemester();
         case 4:
            return mahasiswa.getIpk();
         default :
            return null;
      }
   }

   @Override
```

```
    public String getColumnName(int column){
        return columnNames[column];
    }

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

    public void setMahasiswaList(List<modelMahasiswa> mahasiswaList){
        this.mahasiswaList = mahasiswaList;
        fireTableDataChanged();
    }
}
```

Kode di atas adalah kelas modelTableMahasiswa, yang berfungsi sebagai model tabel untuk menampilkan data mahasiswa dalam bentuk tabel di aplikasi GUI berbasis Swing. Kelas ini memperluas AbstractTableModel dan berfungsi sebagai adaptor antara data (list modelMahasiswa) dan tabel di antarmuka pengguna. Berikut adalah Penjelasannya

- **Atribut mahasiswaList dan columnNames**
- **mahasiswaList**
  Menyimpan daftar objek modelMahasiswa yang akan ditampilkan dalam tabel.
- **columnNames**
  Menyimpan nama-nama kolom tabel sebagai string, yang akan digunakan untuk header kolom.
- **Konstruktor modelTableMahasiswa(List<modelMahasiswa> mahasiswaList)**
  Konstruktor ini menerima list modelMahasiswa sebagai parameter, yang merupakan data yang akan ditampilkan dalam tabel.
- **getRowCount()**
  Mengembalikan jumlah baris dalam tabel berdasarkan ukuran mahasiswaList. Digunakan oleh tabel untuk mengetahui jumlah baris yang akan ditampilkan.
- **getColumnCount()**
  Mengembalikan jumlah kolom, yang diambil dari panjang array columnNames.
- **getValueAt(int rowIndex, int columnIndex)**
  Mengambil nilai dari sel tabel yang ditentukan oleh rowIndex (baris) dan columnIndex (kolom). Nilai yang dikembalikan sesuai dengan atribut modelMahasiswa yang bersangkutan, misalnya ID, NPM, Nama, Semester, dan IPK.
- **getColumnName(int column)**
  Mengembalikan nama kolom sesuai dengan indeksnya, yang diambil dari array columnNames.
- **isCellEditable(int rowIndex, int columnIndex)**
  Menentukan apakah sel tertentu dapat diedit. Di sini, semua sel diatur tidak dapat diedit (false).
- **setMahasiswaList(List<modelMahasiswa> mahasiswaList)**
  Mengatur ulang mahasiswaList dengan list baru dan memicu pembaruan tampilan tabel (fireTableDataChanged()), sehingga tabel akan diperbarui dengan data terbaru.

E.mahasiswaView.java

```
package com.mahasiswa.view;

import com.mahasiswa.model.modelTableMahasiswa;
import com.mahasiswa.model.modelMahasiswa;
import com.mahasiswa.controller.mahasiswaController;
import com.mahasiswa.model.hibernateUtil;
import java.util.List;
import javax.swing.*;

public class mahasiswaView extends javax.swing.JFrame {
    private mahasiswaController controller;

    public void loadMahasiswaTable(){
        List<modelMahasiswa> listMahasiswa = controller.getAllMahasiswa();

        modelTableMahasiswa tableModel = new modelTableMahasiswa(listMahasiswa);

        dataTable.setModel(tableModel);
    }

    public mahasiswaView() {
        initComponents();
        controller = new mahasiswaController();
        hibernateUtil.testConnection();
        loadMahasiswaTable();
    }

    public void clearTextField(){
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        jTextField4.setText("");
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
        String npm = jTextField1.getText();
        String nama = jTextField2.getText();
        int semester = Integer.parseInt(jTextField3.getText());
        float ipk = Float.parseFloat(jTextField4.getText());
        modelMahasiswa mahasiswa = new modelMahasiswa(0, npm, nama, semester, ipk);
        System.out.println(mahasiswa.getIpk());
        System.out.println(mahasiswa.getNama());
        System.out.println(mahasiswa.getSemester());
        System.out.println(mahasiswa.getNpm());

        controller.addMhs(mahasiswa);
        loadMahasiswaTable();
        clearTextField();
    }
```

```
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
      JTextField idField = new JTextField(5);

      JPanel panel = new JPanel();
      panel.add(new JLabel("Masukan ID yang ingin dihapus :"));
      panel.add(idField);

      int result = JOptionPane.showConfirmDialog(null, panel, "Hapus Mahasiswa",
JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

      if(result == JOptionPane.OK_OPTION){
         try{
            int id = Integer.parseInt(idField.getText());
            controller.deleteMhs(id);
            JOptionPane.showMessageDialog(null, "Data berhasil dihapus", "Sukses",
JOptionPane.INFORMATION_MESSAGE);
            loadMahasiswaTable();
         }catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null, "ID harus berupa angka", "Error",
JOptionPane.ERROR_MESSAGE);
         }
      }
   }
```

Kode di atas adalah kelas mahasiswaView, yang merupakan antarmuka pengguna (GUI) berbasis Swing untuk mengelola data mahasiswa. Kelas ini menampilkan data dalam tabel dan memungkinkan pengguna menambahkan atau menghapus data mahasiswa melalui Interface.

- **Atribut dan Konstruktor mahasiswaView()**
- **controller**
  Objek dari mahasiswaController yang digunakan untuk mengelola operasi database pada data mahasiswa.
- **Konstruktor mahasiswaView()**
  Memanggil initComponents() untuk inisialisasi komponen GUI, membuat instance controller, menguji koneksi ke database dengan hibernateUtil.testConnection(), dan memuat data mahasiswa ke dalam tabel dengan loadMahasiswaTable().
- **loadMahasiswaTable()**
  Mengambil data mahasiswa dari database menggunakan controller.getAllMahasiswa() dan mengisi tabel GUI (dataTable) dengan data tersebut menggunakan model modelTableMahasiswa.
- **clearTextField()**
  Metode ini mengosongkan semua kolom teks yang digunakan untuk memasukkan data mahasiswa.
- **Tombol Tambah Mahasiswa (jButton1ActionPerformed)**
- Mengambil data dari kolom teks untuk npm, nama, semester, dan ipk.
- Membuat objek modelMahasiswa dengan data yang diinput, lalu menambahkan mahasiswa baru ini ke database melalui controller.addMhs(mahasiswa).
- Memuat ulang tabel dengan loadMahasiswaTable() dan mengosongkan kolom input dengan clearTextField() setelah data berhasil ditambahkan.

- **Tombol Hapus Mahasiswa (jButton2ActionPerformed)**
- Membuka dialog input menggunakan JOptionPane untuk meminta pengguna memasukkan ID mahasiswa yang ingin dihapus.
- Jika pengguna menekan OK, ID yang dimasukkan akan dikonversi menjadi integer dan dipakai untuk menghapus data mahasiswa dari database melalui controller.deleteMhs(id).
- Jika data berhasil dihapus, tabel akan diperbarui dengan loadMahasiswaTable().

F.pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mahasiswa</groupId>
  <artifactId>RPL2_Pert4_50421535</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>22</maven.compiler.source>
    <maven.compiler.target>22</maven.compiler.target>
    <exec.mainClass>com.mahasiswa.view.RPL2_Pert4_50421535</exec.mainClass>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.hibernate.orm</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>6.6.0.Final</version>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.33</version>
    </dependency>
  </dependencies>

  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <filtering>false</filtering>
      </resource>
    </resources>
  </build>
  <name>RPL2_Pert4_50421535</name>
</project>
```

- D**ependencies**
- **Hibernate Core** (hibernate-core): Menyediakan fungsionalitas ORM untuk interaksi dengan database.
- **MySQL Connector** (mysql-connector-java): Driver untuk koneksi Java dengan database MySQL.
- **Build Resources**
- Menentukan direktori src/main/resources sebagai tempat penyimpanan resource (dalam hal ini, file konfigurasi Hibernate hibernate.cfg.xml) yang akan dimuat ke dalam classpath.

G.hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http:/hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
      <!-- Database Connection Settings -->
      <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
      <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_db</property>
      <property name="hibernate.connection.username">root</property>
      <property name="hibernate.connection.password"></property>

      <!-- JDBC connection pool settings -->
      <property name="hibernate.c3p0.min_size">5</property>
      <property name="hibernate.c3p0.max_size">20</property>
      <property name="hibernate.c3p0.timeout">300</property>
      <property name="hibernate.c3p0.max_statements">50</property>
      <property name="hibernate.c3p0.idle_text_period">3000</property>

      <!-- SQL dialect -->
      <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

      <!-- Echo all executed SQL to stdout -->
      <property name="hibernate.show_sql">true</property>

      <!-- Drop and re-create the database schema on startup -->
      <property name="hibernate.hbm2ddl.auto">update</property>

      <!-- Mapping class -->
      <mapping class="com.mahasiswa.model.modelMahasiswa"/>
   </session-factory>
</hibernate-configuration>

<root>

</root>
```

- **Pengaturan Koneksi Database**
- **hibernate.connection.driver_class**: Menetapkan driver JDBC yang digunakan (com.mysql.cj.jdbc.Driver), yaitu driver MySQL.
- **hibernate.connection.url**: URL koneksi untuk mengakses database (jdbc:mysql://localhost:3306/hibernate_db), di mana localhost adalah alamat server, 3306

adalah port MySQL, dan hibernate_db adalah nama database.

- **hibernate.connection.username** dan **hibernate.connection.password**: Menetapkan kredensial untuk mengakses database. Di sini, penggunaannya adalah root tanpa kata sandi.
- **Pengaturan Pooling Koneksi (C3P0)**
- **hibernate.c3p0.min_size** dan **hibernate.c3p0.max_size**: Menetapkan jumlah minimum (5) dan maksimum (20) koneksi dalam pool.
- **hibernate.c3p0.timeout**: Menetapkan waktu tunggu (dalam detik) sebelum koneksi idle ditutup.
- **hibernate.c3p0.max_statements**: Menentukan jumlah maksimum statement SQL yang dapat disimpan di cache.
- **hibernate.c3p0.idle_test_period**: Menetapkan interval (3000 detik) untuk menguji koneksi idle dalam pool.
- **Pengaturan Dialek SQL**
- **hibernate.dialect**: Menentukan dialek SQL untuk MySQL (org.hibernate.dialect.MySQLDialect), agar Hibernate dapat menghasilkan query SQL yang kompatibel dengan MySQL.
- **Pengaturan Tampilan SQL dan Skema Database**
- **hibernate.show_sql**: Menampilkan query SQL yang dijalankan oleh Hibernate di konsol (diatur ke true).
- **hibernate.hbm2ddl.auto**: Mengatur pengelolaan skema database. Nilai update berarti Hibernate akan otomatis memperbarui skema tanpa menghapus data (jika ada perubahan pada model).
- **Mapping Kelas**
- **<mapping class="com.mahasiswa.model.modelMahasiswa"/>**: Menetapkan kelas modelMahasiswa sebagai entitas yang dipetakan ke dalam tabel database oleh Hibernate. Setiap perubahan pada entitas ini akan tercermin pada tabel terkait.

H.Output