PRAKTIKUM 4

PEMROGRAMAN BERORIENTASI OBJEK

Dosen Pengampu :

Bayu Adhi Nugroho, Ph.D.

Disusun Oleh:

Yuna Ikbar Zaidan (09010622015)

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN AMPEL

SURABAYA

2023

1. TUJUAN

Berikut adalah tujuan dari modul ini:

- Mahasiswa dapat memahami konsep dasar pemrograman berorientasi objek pada subjek pembahasan interface dan implement.
- Mahasiswa mampu mengimplementasikan konsep dasar pemrograman berorientasi objek pada subjek pembahasan interface dan implement.
- Mengetahui keunggulan dari penggunaan interface dan implement.
- Menjadikan pembuatan program dengan bahasa Java sebagai metode belajar.
- Menjadikan program untuk sarana pemahaman dasar dan persiapan modul selanjutnya.

2. DASAR TEORI

Berikut adalah tujuan dari modul ini:

- Java

  Java adalah bahasa yang dikembangkan oleh James Gosling pada tahun 1990-an. Java lahir sebagai bahasa yang dapat berjalan di banyak platform tanpa kompilasi ulang. Berdasarkan Indeks Komunitas Pemrograman TIOBE, Java masih menjadi salah satu bahasa pemrograman paling popular di dunia. Oracle mengatakan 90 persen dari perusahaan Fortune 500 menggunakan Java. Selain itu, Java juga dapat digunakan untuk mengembangkan aplikasi untuk platform desktop, web, mobile, embedded dan IoT.

- Pemrograman Berorientasi Objek

  PBO adalah metode pemrograman berorientasi objek yang bertujuan untuk membuat pengembangan perangkat lunak lebih mudah. OOP memiliki variable dan fungsi yang dibungkus dengan objek atau kelas. Keduanya dapat saling berinteraksi untuk membentuk sebuah program.

- Kelas

  Kelas dapat direpresentasikan sebagai cetak biru, prototipe, atau pabrik yang membuat objek. Dalam membuat nama kelas harus disesuaikan dengan objek yang akan dibuat. Penulisan nama *class* memiliki aturan. Yakni dengan format PascalCase yaitu penulisan nama variable tersusun dari dua kata atau lebih maka tidak perlu diberi spasi di antaranya dan diawali dengan huruf kapital pula.

- Objek

  Ide dasar pada OOP adalah mengkombinasi data dan fungsi untuk mengakses data menjadi sebuah kesatuan unit yang dikenal dengan nama objek. Objek adalah struktur

data yang terdiri dari bidang data dan metode Bersama dengan interaksi mereka untuk merancang aplikasi dan program computer. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

- Enkapsulasi

  Enkapsulasi merupakan sebuah metode menyembunyikan suatu data dalam kelas untuk mencegah semua yang ada di luar kelas dapat mengakses data tersebut secara langsung. Kelas lain dapat mengakses data melalui method yang disediakan dengan nama setter dan getter. Getter berguna untuk membaca data / read only sedangkan setter untuk menulis / write only. Setter dan getter dapat diakses oleh kelas lain.
- Perbedaan antara modifier public, protected, dan private pada method dan variable adalah sebagai berikut:
  a. Public dapat diakses dimanapun.
  b. Private tidak dapat diakses dimanapun kecuali oleh method-method yang ada dalam kelas itu sendiri.
  c. Protected dapat diakses oleh method-method yang sepaket.
- Perbedaan aksesibilitas dari public, protected, dan private adalah sebagai berikut:

Aksesabilitas	public	private	protected
Dari kelas yang sama	Ya	Ya	Ya
Dari sembarang kelas dalam paket yang sama	Ya	Tidak	Ya
Dari sembarang kelas di luar paket	Ya	Tidak	Tidak
Dari subkelas dalam paket yang sama	Ya	Tidak	Ya
Dari subkelas di luar paket	Ya	Tidak	Ya

- Super digunakan untuk merujuk pada kelas dasar (superclass) dari kelas saat ini dalam hierarki pewarisan (inheritance), cara kerjanya adalah mengakses metode atau variable yang ada di kelas dasar saat nama yang sama terdapat pada kelas turunan.
- This digunakan untuk merujuk pada objek saat ini dari kelas saat ini, cara kerjanya adalah membedakan ketika terdapat nama variable local dan nama parameter atau variable anggota kelas yang sama.
- Interface adalah sebuah kontrak yang digunakan untuk mendefinisikan metode-metode yang harus diimplementasikan oleh kelas-kelas lain. Interface hanya mendefinisikan tipe dan nama metode, tetapi tidak memberikan implementasi konkret dari metode-metode tersebut.

- Implementasi merujuk pada tindakan sebenarnya dalam kode untuk mengimplementasikan metode-metode yang didefinisikan dalam sebuah interface. Kelas yang mengimplementasikan sebuah interface harus memberikan implementasi konkret untuk semua metode yang ada dalam interface tersebut.
- Statement dan PreparedStatement adalah dua interface yang digunakan untuk mengirim query SQL ke database.
- Statement digunakan untuk menjalankan pernyataan SQL statis, yaitu hanya dengan parameter yang bersifat fixed, sehingga tidak begitu fleksibel apabila terdapat perubahan query SQL kompleks dengan parameter. Statement dikompilasi setiap kali dieksekusi, sehingga dapat menyebabkan kinerja yang kurang efisien jika sering dilakukan eksekusi. Statement memiliki resiko SQL injection jika data tidak divalidasi dengan benar.
- PreparedStatement digunakan untuk menjalankan pernyataan SQL dinamis dan dapat memiliki parameter yang diisi dengan nilai berbeda-beda, sehingga cocok unuk menghindari SQL injection. Query SQL pada PreparedStatement dikompilasi sekali dan kemudian dapat dieksekusi berkali-kali untuk meningkatkan efisiensi dan kinerja aplikasi. PreparedStatement mendukung parameter binding dengan menggunakan tanda tanya (?) sebagai placeholder untuk parameter dalam pernyataan SQL dan kemudian mengikatkan nilai-nilai parameter ke pernyataan sebelum eksekusi. Parameter binding dilakukan secara urut, sehingga diperlukan penempatan nilai masukan yang tepat supaya tidak terjadi error.

3. TUGAS PRAKTIKUM

Sebelum melakukan proses penulisan kode, perlu masukan library PostgreSQL kedalam project yang akan dijalankan, cara memasukkannya adalah dengan klik kanan pada Libraries project, lalu Add Library, untuk NetBeans versi 18, library PostgreSQL sudah ada sehingga hanya perlu dipilih dan klik OK. Setelah itu buat class baru pada package sehingga muncul jendela baru yang dapat digunakan untuk menulis kode.

Tahap awal yang perlu dilakukan adalah melakukan import untuk java.sql.Connection, java.sql.DriverManager, java.sql.PreparedStatement, java.sql.ResultSet, java.sql.SQLException, dan java.sql.Statement yang dituliskan sebelum blok kode class. Apabila tidak ingin merinci package pada library yang ingin di import, cukup dituliskan java.sql.*, maka otomatis semua bagian dari package java.sql akan ter-import.

```java
package ConnectDB;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

Selanjutnya adalah menuliskan variable yang berisi informasi untuk proses koneksi ke database di dalam blok kode class, variable yang diperlukan antara lain URL/alamat database yang akan diakses, username database, dan password database.

```java
public class DatabaseConnection {

    // Belum membuat close connection
    // Informasi koneksi database
    private static final String DB_URL = "jdbc:postgresql://localhost:5432/DBL_A_M4_09010622015";
    private static final String DB_USER = "postgres";
    private static final String DB_PASSWORD = "20040228";
```

Selanjutnya adalah membuat method untuk koneksi ke database dengan mengandung exception.

```java
// Metode untuk membuat koneksi ke database
public static Connection connect() throws SQLException {
    return DriverManager.getConnection(url:DB_URL, user:DB_USER, password:DB_PASSWORD);
}
```

Selanjutnya adalah membuat method untuk melakukan SELECT dari table mata_kuliah.

```java
// Metode untuk melakukan SELECT dari tabel mata kuliah
public static void selectData() {
    try (Connection connection = connect(); Statement statement = connection.createStatement()) {
        String sql = "SELECT * FROM mata_kuliah";
        ResultSet resultSet = statement.executeQuery(string: sql);

        while (resultSet.next()) {
            String kodeMataKuliah = resultSet.getString(string: "kode_mata_kuliah");
            String namaMataKuliah = resultSet.getString(string: "nama_mata_kuliah");
            int sks = resultSet.getInt(string: "sks");
            int semester = resultSet.getInt(string: "semester");
            String kodeJenis = resultSet.getString(string: "kodejenis");
            System.out.println("Kode Mata Kuliah: " + kodeMataKuliah);
            System.out.println("Nama Mata Kuliah: " + namaMataKuliah);
            System.out.println("Bobot SKS: " + sks);
            System.out.println("Semester: " + semester);
            System.out.println("Jenis Mata Kuliah: " + kodeJenis + "\n");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Selanjutnya adalah membuat method untuk melakukan INSERT ke dalam table mata_kuliah.

```java
// Metode untuk melakukan INSERT data ke tabel mata kuliah
public static void insertData(String kodeMataKuliah, String namaMataKuliah, int sks, int semester, String kodeJenis) {
    try (Connection connection = connect(); PreparedStatement preparedStatement = connection.prepareStatement(
            string: "INSERT INTO mata_kuliah (kode_mata_kuliah, nama_mata_kuliah, sks, semester, kodejenis) VALUES (?, ?, ?, ?, ?)")) {

        preparedStatement.setString(i: 1, string: kodeMataKuliah);
        preparedStatement.setString(i: 2, string: namaMataKuliah);
        preparedStatement.setInt(i: 3, i1: sks);
        preparedStatement.setInt(i: 4, i1: semester);
        preparedStatement.setString(i: 5, string: kodeJenis);
        preparedStatement.executeUpdate();

        System.out.println(x: "Data berhasil ditambahkan!");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Selanjutnya adalah membuat method untuk melakukan UPDATE data pada table mata_kuliah, proses UPDATE dilakukan berdasarkan kode_mata_kuliah.

```java
// Metode untuk melakukan UPDATE data di tabel mata kuliah berdasarkan kode mata kuliah
public static void updateData(String kodeMataKuliah, String namaMataKuliah, int sks, int semester, String kodeJenis) {
    try (Connection connection = connect(); PreparedStatement preparedStatement = connection.prepareStatement(
            string: "UPDATE mata_kuliah SET nama_mata_kuliah = ?, sks = ?, semester = ?, kodejenis = ? WHERE kode_mata_kuliah = ?")) {

        preparedStatement.setString(i: 1, string: namaMataKuliah);
        preparedStatement.setInt(i: 2, i1: sks);
        preparedStatement.setInt(i: 3, i1: semester);
        preparedStatement.setString(i: 4, string: kodeJenis);
        preparedStatement.setString(i: 5, string: kodeMataKuliah);
        preparedStatement.executeUpdate();

        System.out.println(x: "Data berhasil diperbarui!");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Selanjutnya adalah membuat method untuk melakukan DELETE data pada table mata_kuliah, proses DELETE dilakukan berdasarkan kode_mata_kuliah.

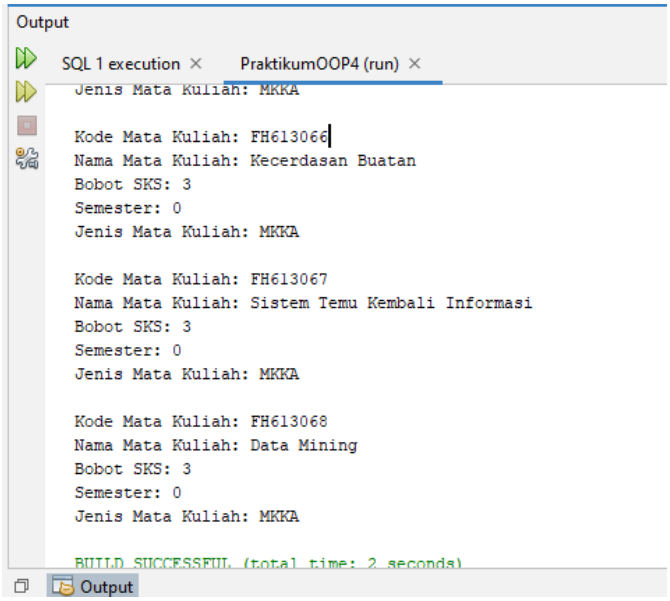
```java
// Metode untuk melakukan DELETE data di tabel mata kuliah berdasarkan kode mata kuliah
public static void deleteData(String kodeMataKuliah) {
    try (Connection connection = connect(); PreparedStatement preparedStatement = connection.prepareStatement(
            string: "DELETE FROM mata_kuliah WHERE kode_mata_kuliah = ?")) {

        preparedStatement.setString(i: 1, string: kodeMataKuliah);
        preparedStatement.executeUpdate();

        System.out.println(x: "Data berhasil dihapus!");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

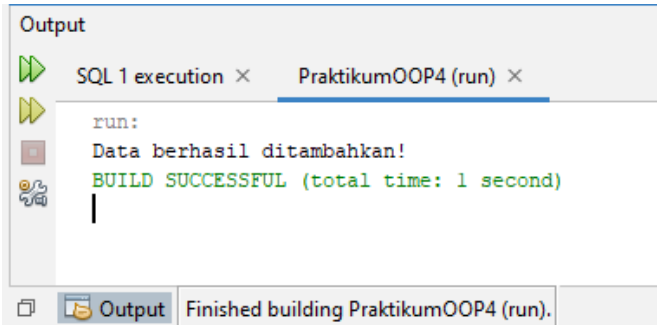
Berikut adalah output dari method SELECT.

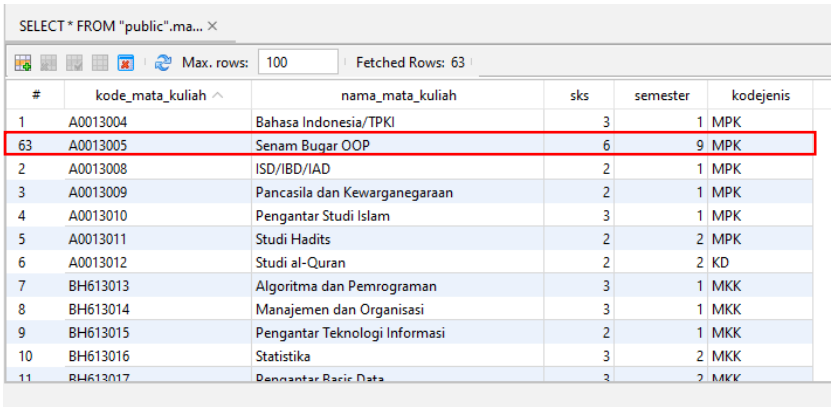
```java
    public static void main(String[] args) {
        // Contoh penggunaan:
        selectData();
//          insertData("", "", , ,"");
//          updateData(1, "Nama Dosen Terbaru", "Jabatan Dosen Terbaru");
//          deleteData(1);
    }
}
```



Berikut adalah output dari method INSERT.

```
    public static void main(String[] args) {
        // Contoh penggunaan:
//        selectData();
        insertData(kodeMataKuliah: "A0013005", namaMataKuliah: "Senam Bugar OOP", sks:6, semester: 9,kodeJenis: "MPK");
//        updateData(1, "Nama Dosen Terbaru", "Jabatan Dosen Terbaru");
//        deleteData(1);
    }
}
```





#	kode_mata_kuliah	nama_mata_kuliah	sks	semester	kodejenis
1	A0013004	Bahasa Indonesia/TPKI	3	1	MPK
63	A0013005	Senam Bugar OOP	6	9	MPK
2	A0013008	ISD/IBD/IAD	2	1	MPK
3	A0013009	Pancasila dan Kewarganegaraan	2	1	MPK
4	A0013010	Pengantar Studi Islam	3	1	MPK
5	A0013011	Studi Hadits	2	2	MPK
6	A0013012	Studi al-Quran	2	2	KD
7	BH613013	Algoritma dan Pemrograman	3	1	MKK
8	BH613014	Manajemen dan Organisasi	3	1	MKK
9	BH613015	Pengantar Teknologi Informasi	2	1	MKK
10	BH613016	Statistika	3	2	MKK
11	BH613017	Pengantar Basis Data	3	2	MKK

Berikut adalah output dari method UPDATE.

```
    public static void main(String[] args) {
        // Contoh penggunaan:
//          selectData();
//          insertData("A0013005", "Senam Bugar OOP", 6, 9,"MPK");
          updateData(kodeMataKuliah: "A0013005", namaMataKuliah: "Kultum (Koding Lulus Tujuh Menit)", sks:1, semester:1,kodeJenis: "MKK");
//          deleteData(1);
    }
}
```

```
Output
SQL 1 execution ×   PraktikumOOP4 (run) ×
run:
Data berhasil diperbarui!
BUILD SUCCESSFUL (total time: 1 second)
Output   Finished building PraktikumOOP4 (run).
```

SELECT * FROM "public".ma... ×

Max. rows: 100 Fetched Rows: 63

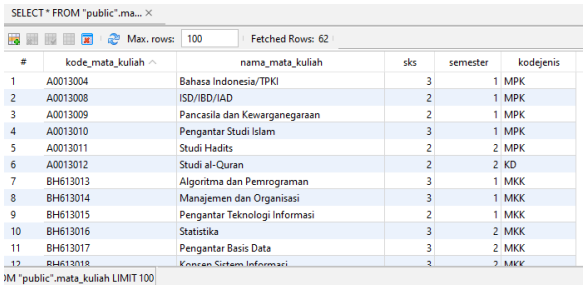
#	kode_mata_kuliah	nama_mata_kuliah	sks	semester	kodejenis
1	A0013004	Bahasa Indonesia/TPKI	3	1	MPK
63	A0013005	Kultum (Koding Lulus Tujuh Menit)	1	1	MKK
2	A0013008	ISD/IBD/IAD	2	1	MPK
3	A0013009	Pancasila dan Kewarganegaraan	2	1	MPK
4	A0013010	Pengantar Studi Islam	3	1	MPK
5	A0013011	Studi Hadits	2	2	MPK
6	A0013012	Studi al-Quran	2	2	KD
7	BH613013	Algoritma dan Pemrograman	3	1	MKK
8	BH613014	Manajemen dan Organisasi	3	1	MKK
9	BH613015	Pengantar Teknologi Informasi	2	1	MKK
10	BH613016	Statistika	3	2	MKK
11	BH613017	Pengantar Basic Data	3	2	MKK

OM "public".mata_kuliah LIMIT 100

Berikut adalah output dari method DELETE.

```
    public static void main(String[] args) {
        // Contoh penggunaan:
//          selectData();
//          insertData("A0013005", "Senam Bugar OOP", 6, 9,"MPK");
//          updateData("A0013005", "Kultum (Koding Lulus Tujuh Menit)", 1, 1,"MKK");
          deleteData(kodeMataKuliah: "A0013005");
    }
}
```

```
ut
SQL 1 execution ×   PraktikumOOP4 (run) ×
run:
Data berhasil dihapus!
BUILD SUCCESSFUL (total time: 1 second)
Output   Finished building PraktikumOOP4 (run).
```



SELECT * FROM "public".ma... ×

Max. rows: 100 Fetched Rows: 62

#	kode_mata_kuliah	nama_mata_kuliah	sks	semester	kodejenis
1	A0013004	Bahasa Indonesia/TPKI	3	1	MPK
2	A0013008	ISD/IBD/IAD	2	1	MPK
3	A0013009	Pancasila dan Kewarganegaraan	2	1	MPK
4	A0013010	Pengantar Studi Islam	3	1	MPK
5	A0013011	Studi Hadits	2	2	MPK
6	A0013012	Studi al-Quran	2	2	KD
7	BH613013	Algoritma dan Pemrograman	3	1	MKK
8	BH613014	Manajemen dan Organisasi	3	1	MKK
9	BH613015	Pengantar Teknologi Informasi	2	1	MKK
10	BH613016	Statistika	3	2	MKK
11	BH613017	Pengantar Basis Data	3	2	MKK
12	BH613018	Konsep Sistem Informasi	3	2	MKK

)M "public".mata_kuliah LIMIT 100

4. KESIMPULAN

Berdasarkan tujuan praktikum yang telah dicapai, dapat disimpulkan bahwa:

Praktikum ini berhasil memberikan pemahaman yang komprehensif kepada mahasiswa mengenai konsep dasar pemrograman berorientasi objek, khususnya dalam konteks subjek pembahasan enkapsulasi. Mahasiswa telah menginternalisasi konsep enkapsulasi dan mampu menerapkannya secara langsung dalam praktik melalui implementasi program. Proses pembelajaran ini mengungkapkan keunggulan signifikan dari penggunaan enkapsulasi dalam mengorganisir kode dan data, serta melindungi integritas informasi.

Praktikum ini juga berhasil menciptakan lingkungan belajar yang efektif melalui penggunaan bahasa pemrograman Java. Dengan melibatkan mahasiswa dalam pembuatan program, praktikum ini telah membantu memperdalam pemahaman konsep-konsep dasar, sekaligus memberikan dasar yang kuat untuk modul pembelajaran yang lebih lanjut. Dengan demikian, praktikum ini telah mencapai tujuan untuk menjadikan pembuatan program sebagai metode pembelajaran yang efektif, serta menjadi langkah persiapan yang tepat untuk memahami materi yang lebih kompleks di masa depan.

Secara keseluruhan, praktikum ini telah berhasil memberikan mahasiswa pemahaman yang solid tentang konsep enkapsulasi dalam pemrograman berorientasi objek, mengajarkan mereka bagaimana mengimplementasikan konsep tersebut dalam bahasa Java, serta mengilustrasikan manfaat penting dari penggunaan enkapsulasi. Diharapkan bahwa pemahaman dan keterampilan yang diperoleh dari praktikum ini akan membantu mahasiswa dalam perjalanan pembelajaran mereka yang lebih lanjut.

5. REFERENSI

Smith, J. D. (2020). Understanding Object-Oriented Programming Concepts. Journal of Computer Science Education, 28(3), 245-260.

Johnson, A. R., & Williams, M. L. (2019). Implementing Object Encapsulation in Java: A Practical Approach. Programming Paradigms Quarterly, 14(2), 87-102.

Brown, C. E., & Jones, R. K. (2018). The Benefits of Encapsulation in Software Development. Software Engineering Journal, 26(4), 315-328.

Thompson, L. M. (2021). Effective Learning through Programming: A Case Study Using Java. Education and Technology Research, 40(1), 56-72.

Davis, P. S. (2017). The Role of Practical Programming Exercises in Building Strong Foundations. Journal of Computer Science Education, 23(2), 148-165.