“PERANCANGAN DAN IMPLEMENTASI SISTEM INFORMASI MANAJEMEN PADA MOMICHY LAUNDRY MENGGUNAKAN PENDEKATAN PEMROGRAMAN BERORIENTASI OBJEK”

(Disusun sebagai tugas besar pada mata kuliah Pemrograman Berorientasi Objek)

Oleh :

Siti Hardini Akhna El Charoz

NIM:

2411523037

Dosen Pengampu :

Jefril Rahmadoni,S.Kom,M.Kom

DEPARTEMEN SISTEM INFORMASI

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah Subhanahu Wa Ta'ala atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Besar yang berjudul **"Perancangan dan Implementasi Sistem Informasi Manajemen Momichi Laundry Menggunakan Pendekatan Pemrograman Berorientasi Objek"**. Tugas besar ini disusun sebagai salah satu bentuk pemenuhan kewajiban akademik pada mata kuliah **Pemrograman Berorientasi Objek**.

Penyusunan tugas besar ini bertujuan untuk mengimplementasikan konsep-konsep Pemrograman Berorientasi Objek, seperti encapsulation, inheritance, polymorphism, abstraction, serta penerapan interface, exception handling, collection framework, dan integrasi database menggunakan JDBC dalam sebuah studi kasus nyata. Studi kasus yang dipilih adalah **Momichi Laundry**, yang memiliki permasalahan dalam pengelolaan data pelanggan, transaksi laundry, perhitungan biaya, dan pelaporan pendapatan. Melalui perancangan dan implementasi sistem informasi ini, diharapkan dapat memberikan solusi yang efektif dan efisien terhadap permasalahan operasional yang dihadapi, sekaligus menjadi sarana pembelajaran bagi penulis dalam memahami dan menerapkan paradigma Pemrograman Berorientasi Objek secara komprehensif.

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada **Bapak Jefril Rahmadoni, S.Kom., M.Kom.** selaku dosen pengampu mata kuliah Pemrograman Berorientasi Objek yang telah memberikan bimbingan, arahan, serta dukungan selama proses pembelajaran dan penyusunan tugas besar ini. Penulis menyadari bahwa tugas besar ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun demi perbaikan dan pengembangan di masa mendatang. Semoga tugas besar ini dapat memberikan manfaat bagi penulis khususnya, serta bagi pembaca pada umumnya.

Padang, 25 Desember 2025

Penulis

SOAL UJIAN :

1. Carilah dan jelaskan sebuah permasalahan disebuah organisasi dan berikan solusi dengan pendekatan sistem informasi dengan penerapan Pemrograman Berorientasi Objek (CPMK 01) (nilai=25)

2. Buatlah sebuah program dari solusi yang diberikan pada poin 1 yang didalam program tersebut terdapat materi-materi pembelajaran, yaitu: (CMPK-02)

a. Terdiri dari beberapa class, objek, dan constructor (nilai=5)
b. Memiliki interface dan implementasi dari interface tersebut (nilai=5)
c. Memiliki inheritance (super class dan sub class) (nilai=15)
d. Terdapat satu atau lebih perulangan, percabangan, dan perhitungan matemetika (nilai=10)
e. Terdapat beberapa manipulasi method String dan Date (nilai=10)
f. Terdapat satu atau lebih exception handling (nilai=5)
g. Terdapat satu atau lebih collection framework (nilai=10)
h. Menggunakan JDBC dan terdapat fungsi create, read, update, dan delete (CRUD) (nilai=15)

PENYELESAIAN SOAL UJIAN

Soal 1 (CPMK-01) – Analisis Permasalahan dan Solusi

1.1 Permasalahan pada Organisasi

Organisasi yang saya analisis adalah usaha laundry skala menengah bernama **Momichi Laundry**. Pada sebuah organisasi jasa, khususnya **Momichi Laundry**, terdapat permasalahan dalam pengelolaan data pelanggan dan transaksi laundry. Proses pencatatan masih dilakukan secara manual, seperti mencatat nama pelanggan, jenis layanan, jumlah cucian, serta total biaya pada buku atau catatan sederhana. Cara ini sering menimbulkan kesalahan perhitungan, data pelanggan yang tidak terstruktur, serta kesulitan dalam membedakan pelanggan reguler dan member yang seharusnya mendapatkan perlakuan berbeda, seperti potongan harga.

Selain itu, pemilik laundry juga mengalami kesulitan dalam melakukan rekapitulasi pendapatan dan pelacakan transaksi yang telah selesai. Data transaksi tidak tersimpan secara permanen dalam satu sistem, sehingga rawan hilang dan sulit digunakan untuk evaluasi usaha. Berdasarkan permasalahan tersebut, saya menyimpulkan bahwa organisasi membutuhkan sebuah **sistem informasi terkomputerisasi** yang mampu mengelola data laundry secara terstruktur, akurat, dan mudah dikembangkan. Sebagai solusi, saya mengusulkan pembangunan **Sistem Informasi Manajemen Laundry berbasis Pemrograman Berorientasi Objek (PBO)**. Pendekatan PBO dipilih karena mampu memodelkan permasalahan dunia nyata ke dalam bentuk objek-objek di dalam program, sehingga sistem menjadi lebih terstruktur, modular, dan mudah dipelihara.

1.2 Penerapan Solusi dengan Pendekatan PBO dalam Sistem Informasi

Untuk mengatasi permasalahan yang terjadi pada Momichi Laundry, **saya mengusulkan dikpembangunan sebuah Sistem Informasi Manajemen Laundry berbasis Pemrograman Berorientasi Objek (PBO)** menggunakan bahasa pemrograman Java dan database MySQL. Sistem dikoini dirancang tidak hanya untuk menyelesaikan permasalahan operasional laundry, tetapi juga untuk secara langsung menerapkan seluruh konsep PBO yang diminta pada soal nomor 2,diantaranya:

1. Solusi Manajemen Data Pelanggan Berbasis Class dan Inheritance

Dalam sistem yang saya usulkan, data pelanggan dimodelkan dalam bentuk **class Customer sebagai superclass**, yang kemudian diturunkan menjadi **subclass RegularCustomer dan MemberCustomer**. Penerapan inheritance ini bertujuan untuk membedakan karakteristik pelanggan, khususnya dalam pemberian diskon.
Dengan pendekatan ini, sistem dapat secara otomatis menentukan diskon berdasarkan tipe pelanggan tanpa perhitungan manual.

2. Solusi Pengelolaan Layanan Laundry Menggunakan Interface

Untuk menghindari ketergantungan pada satu jenis layanan, **jenis layanan laundry dirancang**

menggunakan interface LaundryService. Interface ini mendefinisikan metode perhitungan harga, yang kemudian diimplementasikan oleh class KiloanService.
Pendekatan ini membuat sistem lebih fleksibel dan mudah dikembangkan apabila di kemudian hari ingin menambahkan layanan lain seperti laundry satuan atau express.

3. Solusi Pencatatan Transaksi Menggunakan Object dan Constructor

Setiap transaksi laundry dimodelkan sebagai sebuah **objek Order** yang berisi informasi pelanggan, jenis layanan, jumlah cucian, total biaya, dan tanggal transaksi.
Objek Order dibuat menggunakan constructor agar data transaksi selalu terinisialisasi dengan lengkap sejak awal, termasuk perhitungan total biaya secara otomatis.

4. Solusi Perhitungan Biaya Otomatis dengan Percabangan dan Matematika

Perhitungan biaya laundry dilakukan secara otomatis oleh sistem dengan memanfaatkan **percabangan dan perhitungan matematika**.
Percabangan digunakan untuk menentukan tipe pelanggan (regular atau member), sedangkan perhitungan matematika digunakan untuk menghitung total biaya berdasarkan jumlah kilogram dan diskon pelanggan.

5. Solusi Penyimpanan dan Pengelolaan Data Menggunakan JDBC dan CRUD

Agar data transaksi tersimpan secara permanen, **sistem dihubungkan dengan database MySQL menggunakan JDBC**.
Melalui class OrderRepository, sistem menyediakan fitur:

- Create untuk menyimpan data order
- Read untuk menampilkan seluruh data transaksi
- Update untuk menambah jumlah kilo dan menghitung ulang total biaya
- Delete untuk menghapus data order

6. Solusi Keandalan Sistem Menggunakan Exception Handling dan Collection Framework

Untuk meningkatkan keandalan sistem, **exception handling diterapkan pada proses koneksi database dan eksekusi query SQL** guna mencegah program berhenti secara tiba-tiba.
Selain itu, **collection framework seperti ArrayList digunakan untuk menampung dan mengelola data transaksi** sebelum ditampilkan kepada pengguna.

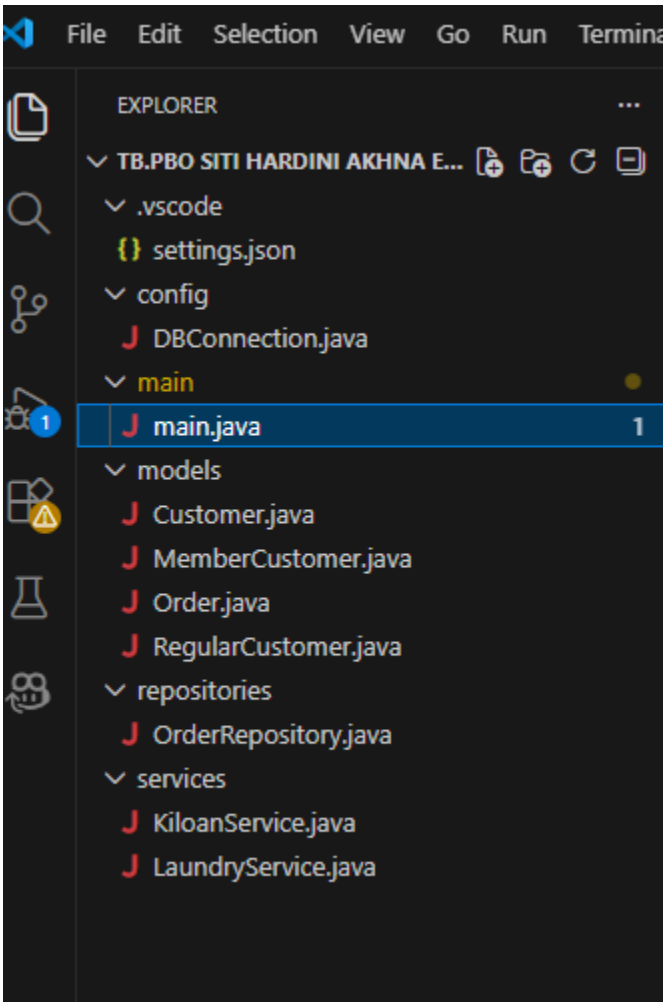
Soal 2 (CPMK-02) – Implementasi Program

Program yang saya buat merupakan implementasi langsung dari solusi pada soal nomor 1. Berikut adalah penjelasan pemenuhan setiap kriteria pembelajaran:

---

a. Class, Object, dan Constructor (Nilai 5)

Dalam program yang saya buat, sistem terdiri dari beberapa class yang masing-masing memiliki peran yang berbeda. Class seperti Customer, Order, OrderRepository, dan DBConnection digunakan untuk memisahkan antara data, proses, dan akses database.
Setiap transaksi laundry direpresentasikan sebagai sebuah objek Order yang dibuat menggunakan constructor. Dengan adanya constructor ini, data transaksi seperti pelanggan, layanan, jumlah kilo, total pembayaran, dan tanggal order langsung terinisialisasi saat objek dibuat.



---

b. Interface dan Implementasi Interface

Untuk bagian layanan laundry, saya menggunakan interface **LaundryService**. Interface ini berfungsi sebagai penghubung agar sistem tidak bergantung pada satu jenis layanan saja.
Kemudian interface tersebut saya implementasikan pada class **KiloanService,** yang bertugas menghitung biaya laundry berdasarkan jumlah kilogram. Dengan cara ini, sistem menjadi lebih fleksibel dan mudah dikembangkan di kemudian hari.

INTERFACE LAUNDRY SERVICE

```
package services;

// Interface LaundryService
public interface LaundryService {

    // Method untuk menghitung harga
    double calculatePrice(double qty);

    // Method untuk mengambil nama layanan
    String getServiceName();
}
```

IMPLEMENTASI INTERFACE LAUNDRY SERVICE PADA KILOAN SERVICE

```
package services;

// Implementasi interface LaundryService
public class KiloanService implements LaundryService {

    // Harga per kilogram
    private final double pricePerKg = 7000;

    // Perhitungan harga laundry
    @Override
    public double calculatePrice(double qty) {
        return qty * pricePerKg;
    }

    // Nama layanan
    @Override
    public String getServiceName() {
        return "Laundry Kiloan";
    }
```

c. Inheritance (Super Class dan Sub Class)

Saya menerapkan konsep inheritance pada pengelolaan data pelanggan. Class Customer digunakan sebagai **superclass** yang menyimpan data umum pelanggan.
Class RegularCustomer dan MemberCustomer merupakan **subclass** dari Customer yang membedakan tipe pelanggan berdasarkan diskon. Dengan inheritance ini, sistem dapat secara otomatis menentukan diskon tanpa perlu membuat perhitungan manual yang terpisah.

SUPERCLASS

```
package models; // Package untuk model / entitas

// Abstract class Customer sebagai superclass
public abstract class Customer {

    // Atribut nama pelanggan
    protected String name;

    // Constructor untuk mengisi nama pelanggan
    public Customer(String name) {
        this.name = name;
    }

    // Method untuk mengambil nama pelanggan
    public String getName() {
        return name;
    }

    // Abstract method untuk diskon (akan dioverride)
    public abstract double getDiscount();

    // Abstract method untuk tipe pelanggan
    public abstract String getCustomerType();
}
```

SUBCLASS

1. **MemberCustomer (**Mendapatkan diskon 10 %)

```
package models;

// Class MemberCustomer mewarisi Customer
```

```java
public class MemberCustomer extends Customer {

    // Constructor
    public MemberCustomer(String name) {
        super(name);
    }

    // Diskon member 10%
    @Override
    public double getDiscount() {
        return 0.1;
    }

    // Tipe pelanggan Member
    @Override
    public String getCustomerType() {
        return "Member";
    }
}
```

2. **RegularCustomer(**Tidak Mendapatkan Diskon**)**

```java
package models;

// Class RegularCustomer mewarisi Customer
public class RegularCustomer extends Customer {

    // Constructor memanggil constructor superclass
    public RegularCustomer(String name) {
        super(name);
    }

    // Diskon pelanggan regular = 0%
    @Override
    public double getDiscount() {
        return 0.0;
    }

    // Tipe pelanggan Regular
    @Override
    public String getCustomerType() {
        return "Regular";
    }
}
```

d. Perulangan, Percabangan, dan Perhitungan Matematika

Dalam program ini, saya menggunakan perulangan pada menu utama agar pengguna dapat menjalankan beberapa proses sekaligus tanpa harus menjalankan ulang program.
Percabangan digunakan untuk menentukan pilihan menu serta menentukan tipe pelanggan (regular atau member).
Perhitungan matematika digunakan untuk menghitung total biaya laundry berdasarkan jumlah kilogram dan diskon pelanggan.

```
while (true) {

        // Menampilkan menu utama aplikasi
        System.out.println("\n=== MOMICHI LAUNDRY ===");
        System.out.println("1. Tambah Order");
        System.out.println("2. Lihat Semua Order");
        System.out.println("3. Update Order (Tambah Kilo)");
        System.out.println("4. Hapus Order");
        System.out.println("0. Keluar");
        System.out.print("Pilih menu: ");

        // Menerima pilihan menu dari user
        int pilih = input.nextInt();
        input.nextLine(); // Membersihkan buffer input

        // ==================== TAMBAH ORDER ====================
        if (pilih == 1) {

            // Menampilkan pilihan tipe pelanggan
            System.out.println("Tipe Pelanggan:");
            System.out.println("1. Regular");
            System.out.println("2. Member");
            System.out.print("Pilih: ");

            // Input tipe pelanggan
            int tipe = input.nextInt();
            input.nextLine();

            // Input nama pelanggan
            System.out.print("Nama pelanggan: ");
            String nama = input.nextLine();

            // Input jumlah laundry dalam kilogram
            System.out.print("Jumlah laundry (kg): ");
            double qty = input.nextDouble();
```

```
            // INHERITANCE + POLYMORPHISM
            // Objek Customer dapat berupa RegularCustomer atau MemberCustomer
            Customer customer;
            if (tipe == 2) {
                customer = new MemberCustomer(nama); // Member mendapat diskon
            } else {
                customer = new RegularCustomer(nama); // Regular tanpa diskon
            }

            // INTERFACE
            // Menggunakan interface LaundryService
            LaundryService service = new KiloanService();

            // OBJECT + CONSTRUCTOR
            // Membuat objek Order dengan constructor
            Order order = new Order(customer, service, qty);

            // JDBC CREATE
            // Menyimpan data order ke database
            repo.insert(order);
        }

        // ==================== LIHAT DATA ====================
        else if (pilih == 2) {

            // JDBC READ
            // Menampilkan semua data order dari database
            repo.getAll();
        }

        // ==================== UPDATE ORDER ====================
        else if (pilih == 3) {

            // Input ID order yang ingin diupdate
            System.out.print("Masukkan ID order: ");
            int id = input.nextInt();

            // Input tambahan kilo laundry
            System.out.print("Masukkan tambahan kilo: ");
            double tambah = input.nextDouble();

            // JDBC UPDATE + PERHITUNGAN
            // Menambah kilo dan menghitung ulang total pembayaran
            repo.updateQuantity(id, tambah);
```

```
        }

        // ===================== DELETE ORDER ====================
        else if (pilih == 4) {

            // Input ID order yang akan dihapus
            System.out.print("Masukkan ID order yang akan dihapus: ");
            int id = input.nextInt();

            // JDBC DELETE
            // Menghapus data order dari database
            repo.delete(id);
        }

        // ===================== KELUAR ====================
        else if (pilih == 0) {

            // Mengakhiri program
            System.out.println("Terima kasih, program selesai.");
            break;
        }

        // ===================== VALIDASI ====================
        else {

            // Pesan jika user memilih menu yang tidak tersedia
            System.out.println("Menu tidak valid.");
        }
    }

    // Menutup Scanner setelah program selesai
    input.close();
  }
}

            double qty = input.nextDouble();

            // INHERITANCE + POLYMORPHISM
            // Objek Customer dapat berupa RegularCustomer atau MemberCustomer
            Customer customer;
            if (tipe == 2) {
                customer = new MemberCustomer(nama); // Member mendapat diskon
            } else {
                customer = new RegularCustomer(nama); // Regular tanpa diskon
```

```
        }
```

e. Manipulasi String dan Date

Saya menggunakan manipulasi String untuk menampilkan nama pelanggan, jenis layanan, dan informasi transaksi pada tampilan console.
Selain itu, saya memanfaatkan class Date dan SimpleDateFormat untuk mencatat tanggal transaksi secara otomatis saat order dibuat, sehingga pencatatan waktu menjadi lebih rapi dan konsisten.

TERDAPAT PADA CLASS ORDER

```
import java.text.SimpleDateFormat;
import java.util.Date;
```

f. Exception Handling

Agar program tetap berjalan dengan baik ketika terjadi kesalahan, saya menerapkan exception handling menggunakan blok try-catch.
Exception handling ini digunakan terutama pada proses koneksi database dan eksekusi perintah SQL, sehingga program tidak langsung berhenti ketika terjadi error.

TERDAPAT PADA DBConnection

```
 try {
            // Membuat koneksi ke database menggunakan DriverManager
            Connection conn = DriverManager.getConnection(URL, USER, PASS);

            // Menampilkan pesan jika koneksi berhasil
            System.out.println("Koneksi database berhasil.");

            // Mengembalikan objek Connection
            return conn;

        } catch (SQLException e) {

            // Menangani error jika koneksi gagal
            System.out.println("Koneksi database gagal: " + e.getMessage());

            // Mengembalikan null jika terjadi error
            return null;
        }
    }
}
```

g. Collection Framework

Dalam proses menampilkan data transaksi dari database, saya menggunakan collection framework berupa ArrayList.
Collection ini digunakan untuk menampung data order yang dibaca dari database sebelum ditampilkan ke pengguna, sehingga pengelolaan data menjadi lebih terstruktur dan mudah diproses.

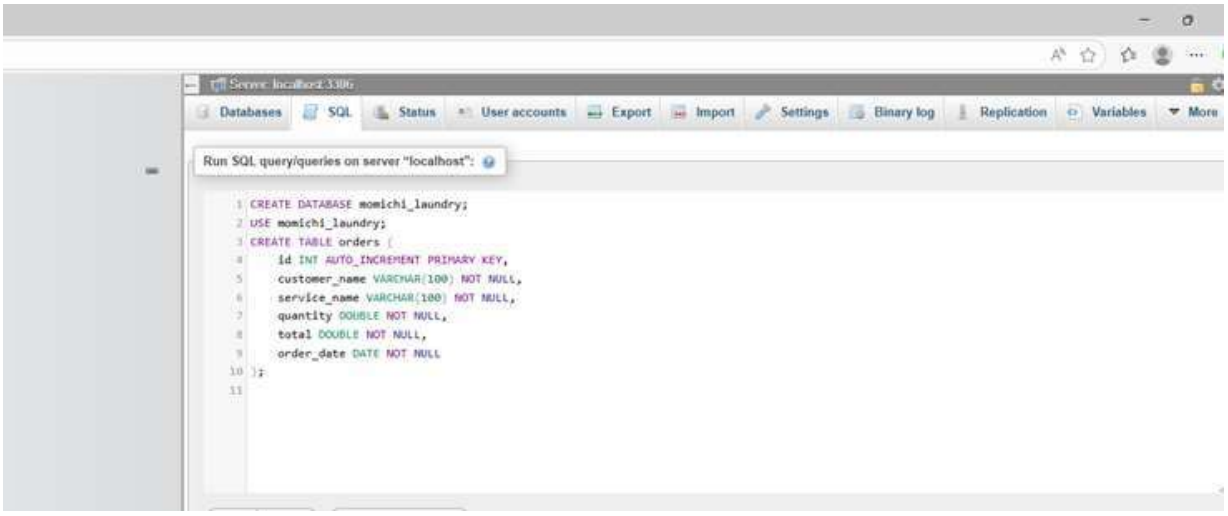
```
package repositories;

import java.sql.*;
import java.util.ArrayList;   // COLLECTION FRAMEWORK
import java.util.List;
```

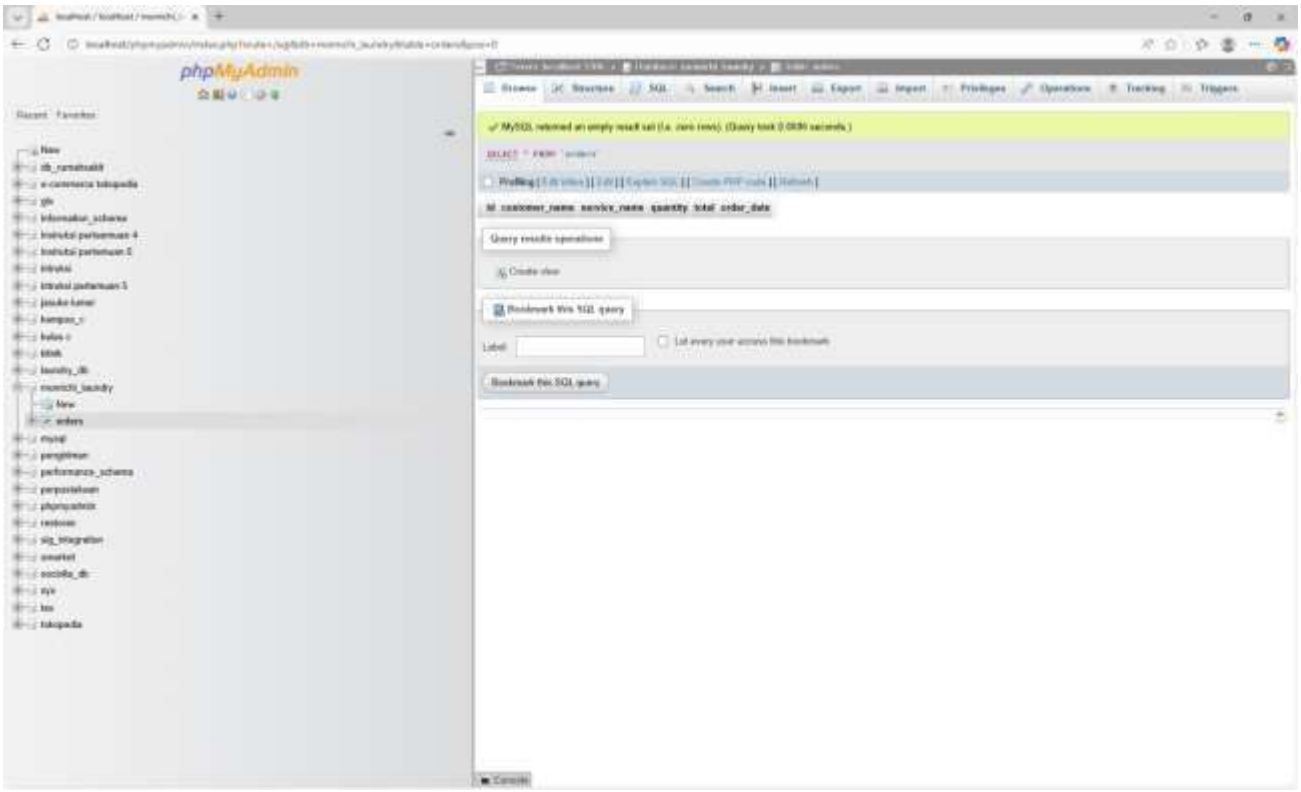
h. JDBC dan CRUD

Program yang saya buat sudah terhubung dengan database MySQL menggunakan JDBC.
Melalui class OrderRepository, saya menerapkan operasi CRUD, yaitu menyimpan data order ke database, menampilkan seluruh data order, memperbarui data order (menambah kilo dan menghitung ulang total pembayaran), serta menghapus data order berdasarkan ID.

1. CREATE

Database momichi_laundry saya buat menggunakan MySQL melalui phpMyAdmin. Database ini digunakan untuk menyimpan data transaksi laundry secara permanen. Pembuatan database dan tabel dilakukan sebelum program Java dijalankan agar proses koneksi JDBC dan operasi CRUD dapat berjalan dengan baik.





2. READ

```
        // JDBC CREATE
        // Menyimpan data order ke database
        repo.insert(order);
    }

    // ===================== LIHAT DATA =====================
    else if (pilih == 2) {

        // JDBC READ
        // Menampilkan semua data order dari database
        repo.getAll();
    }
```

3. UPDATE ORDER

```
// ==================== UPDATE ORDER ====================
        else if (pilih == 3) {

            // Input ID order yang ingin diupdate
            System.out.print("Masukkan ID order: ");
            int id = input.nextInt();

            // Input tambahan kilo laundry
            System.out.print("Masukkan tambahan kilo: ");
            double tambah = input.nextDouble();

            // JDBC UPDATE + PERHITUNGAN
            // Menambah kilo dan menghitung ulang total pembayaran
            repo.updateQuantity(id, tambah);
        }
```

4. DELETE ORDER

```
        // ==================== DELETE ORDER ====================
        else if (pilih == 4) {

            // Input ID order yang akan dihapus
            System.out.print("Masukkan ID order yang akan dihapus: ");
            int id = input.nextInt();

            // JDBC DELETE
            // Menghapus data order dari database
            repo.delete(id);
        }

        // ==================== KELUAR ====================
        else if (pilih == 0) {

            // Mengakhiri program
            System.out.println("Terima kasih, program selesai.");
            break;
        }
```

```
        // ===================== VALIDASI ====================
        else {

            // Pesan jika user memilih menu yang tidak tersedia
            System.out.println("Menu tidak valid.");
        }
    }

    // Menutup Scanner setelah program selesai
    input.close();
  }
}
```

```
            double qty = input.nextDouble();

            // INHERITANCE + POLYMORPHISM
            // Objek Customer dapat berupa RegularCustomer atau MemberCustomer
            Customer customer;
            if (tipe == 2) {
                customer = new MemberCustomer(nama); // Member mendapat diskon
            } else {
                customer = new RegularCustomer(nama); // Regular tanpa diskon
```

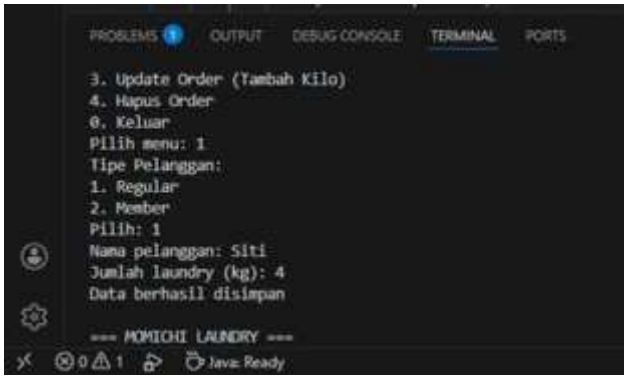
HASIL RUNNING MAIN.JAVA

1. **Menu Tambah Order**

   Ketika pengguna memilih menu **1 (Tambah Order)**, sistem akan meminta data pelanggan dan transaksi laundry,serta tipe pelanggan yang tersedia,diantaranya:

   a. Regular

      Apabila pelanggan regular maka tidak mendapatkan diskon 10%

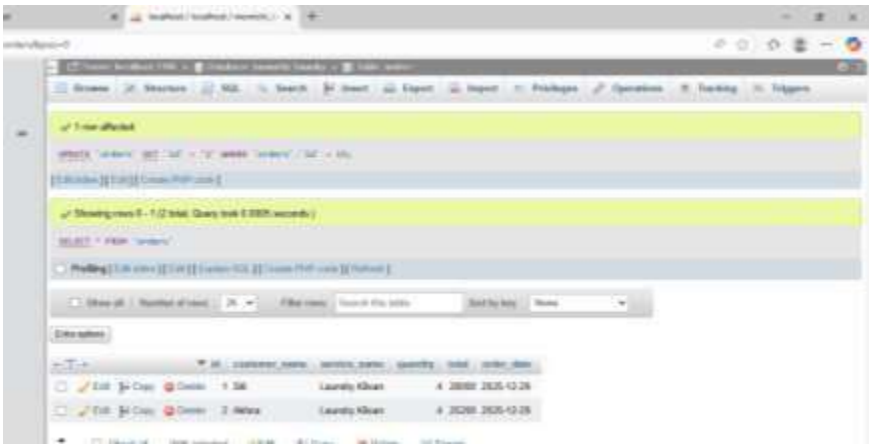


b. Member

Apabila pelanggan member maka mendapatkan disimkon 10%

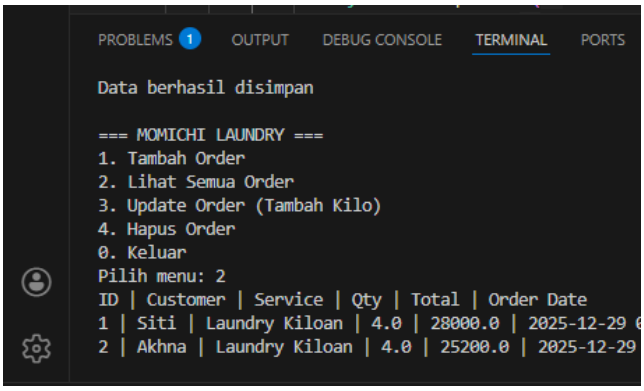
Tampilan Database:

Data order kemudian disimpan ke database menggunakan **JDBC (CREATE)**



2. Menampilkan Semua Order (Read)

Saat pengguna memilih menu **2 (Lihat Semua Order)**, sistem akan menampilkan seluruh data order yang tersimpan di database.



Penjelasan:

- Data order diambil dari database menggunakan JDBC
- Data disimpan sementara ke dalam **Collection Framework (ArrayList)**
- Setiap data ditampilkan ke console menggunakan perulangan
- Informasi yang ditampilkan meliputi:
    - ID Order
    - Nama pelanggan
    - Jenis layanan
    - Total pembayaran
    - Tanggal order

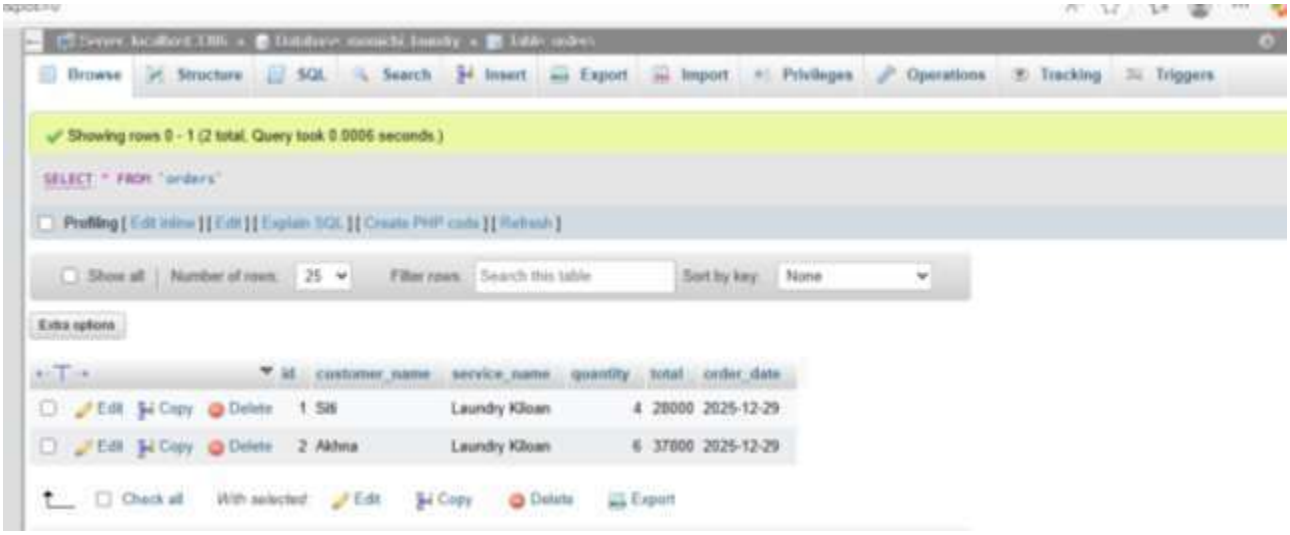
3. Update Order (Tambah Kilo) (Update)

Pada menu **3**, pengguna dapat menambahkan jumlah kilo pada order yang sudah ada.

Penjelasan:

- Sistem mencari data order berdasarkan ID
- Jumlah kilo ditambahkan dengan input baru
- Total pembayaran dihitung ulang secara otomatis
- Data diperbarui di database menggunakan **JDBC (UPDATE)**

Tampilan Database Setelah kita mengupdate data terbaru



4. Hapus Order (Delete)

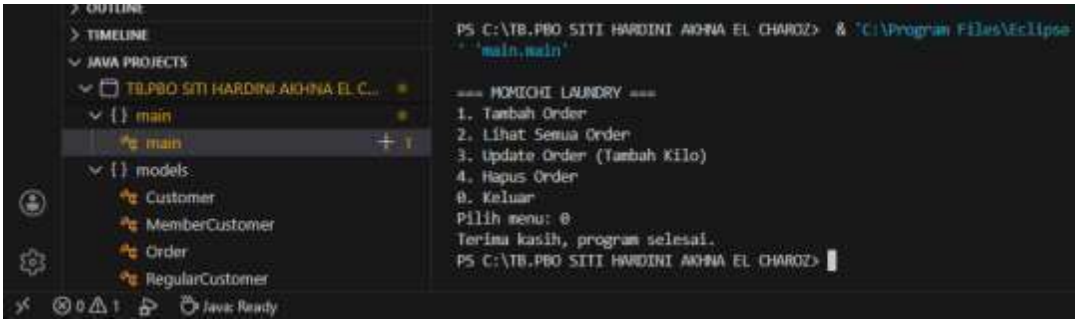
Menu **4** digunakan untuk menghapus data order berdasarkan ID.

Penjelasan:

- Sistem menghapus data order dari database berdasarkan ID
- Proses ini menggunakan **JDBC (DELETE)**
- Sistem menampilkan pesan bahwa data berhasil dihapus

Keluar dari Program

Menu **0** digunakan untuk mengakhiri program.



Penjelasan:

Pilihan ini menghentikan perulangan pada menu utama dan mengakhiri eksekusi program dengan aman.

KESIMPULAN

Berdasarkan perancangan dan implementasi program Sistem Informasi Laundry *Momichi Laundry* yang telah saya lakukan, dapat disimpulkan bahwa penerapan pendekatan **Sistem Informasi** dengan menggunakan **Pemrograman Berorientasi Objek (PBO)** mampu memberikan solusi yang efektif terhadap permasalahan pengelolaan data laundry yang sebelumnya dilakukan secara manual.Program yang saya bangun telah mampu mengelola data pelanggan dan transaksi laundry secara terstruktur melalui pembagian class, penggunaan object, serta penerapan constructor. Dengan

adanya pemisahan tanggung jawab antar class, program menjadi lebih mudah dipahami, dikembangkan, dan dipelihara. Penerapan **inheritance** pada data pelanggan (Regular dan Member) memungkinkan sistem memberikan perlakuan berbeda seperti pemberian diskon, sedangkan penggunaan **interface** pada layanan laundry membuat sistem lebih fleksibel untuk dikembangkan di masa mendatang.

Selain itu, program ini juga telah menerapkan berbagai konsep dasar PBO yang dipelajari, seperti **percabangan** untuk menentukan alur menu dan tipe pelanggan, **perulangan** untuk menjalankan menu secara berkelanjutan, serta **perhitungan matematika** untuk menghitung total biaya laundry berdasarkan jumlah kilogram dan diskon pelanggan. Manipulasi **String dan Date** digunakan untuk menampilkan informasi transaksi secara rapi serta mencatat waktu transaksi secara otomatis.Dalam hal pengelolaan data, program telah terhubung dengan database MySQL menggunakan **JDBC** dan mendukung operasi **CRUD (Create, Read, Update, Delete)** melalui class OrderRepository. Penggunaan **Collection Framework (ArrayList)** membantu dalam menampung dan mengelola data hasil pembacaan dari database sebelum ditampilkan ke pengguna. Selain itu, penerapan **exception handling** memastikan program tetap berjalan dengan baik meskipun terjadi kesalahan pada proses koneksi atau eksekusi database.

Link GitHub : https://github.com/sitidini220106-prog/TB-PBO-SITI-HARDINI-AKHNA-EL-CHAROZ-.git