LAPORAN PRAKTIKUM PEMROGRAMAN BERBASIS OBJEK

PERTEMUAN 4



OLEH:

IRGI FATIHUL IHSAN

NIM: 2311533010

DOSEN PENGAMPU:

NURFIAH, S.ST, M.KOM

FAKULTAS TEKNOLOGI INFORMASI

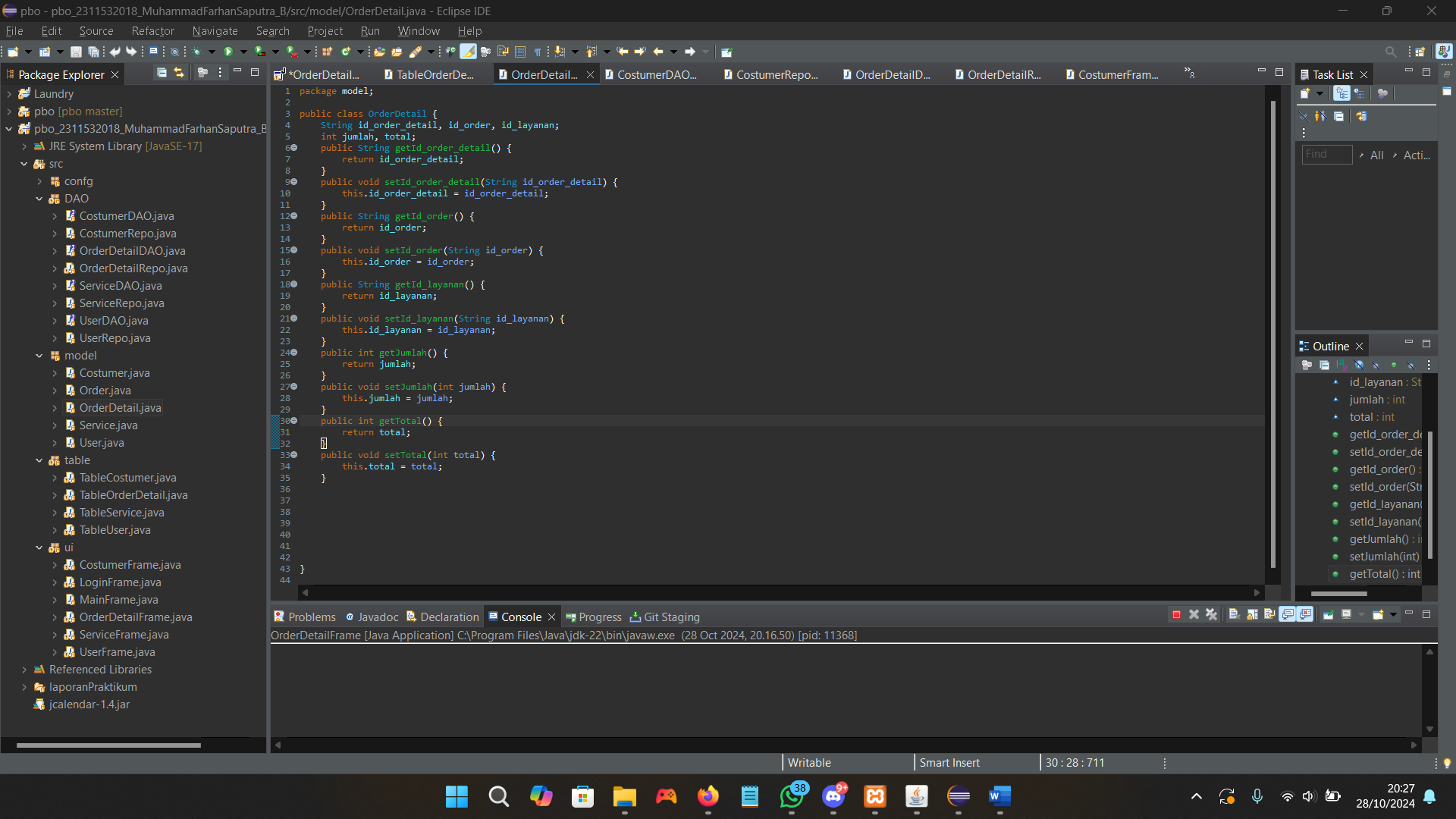
DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG, SEPTEMBER 2024

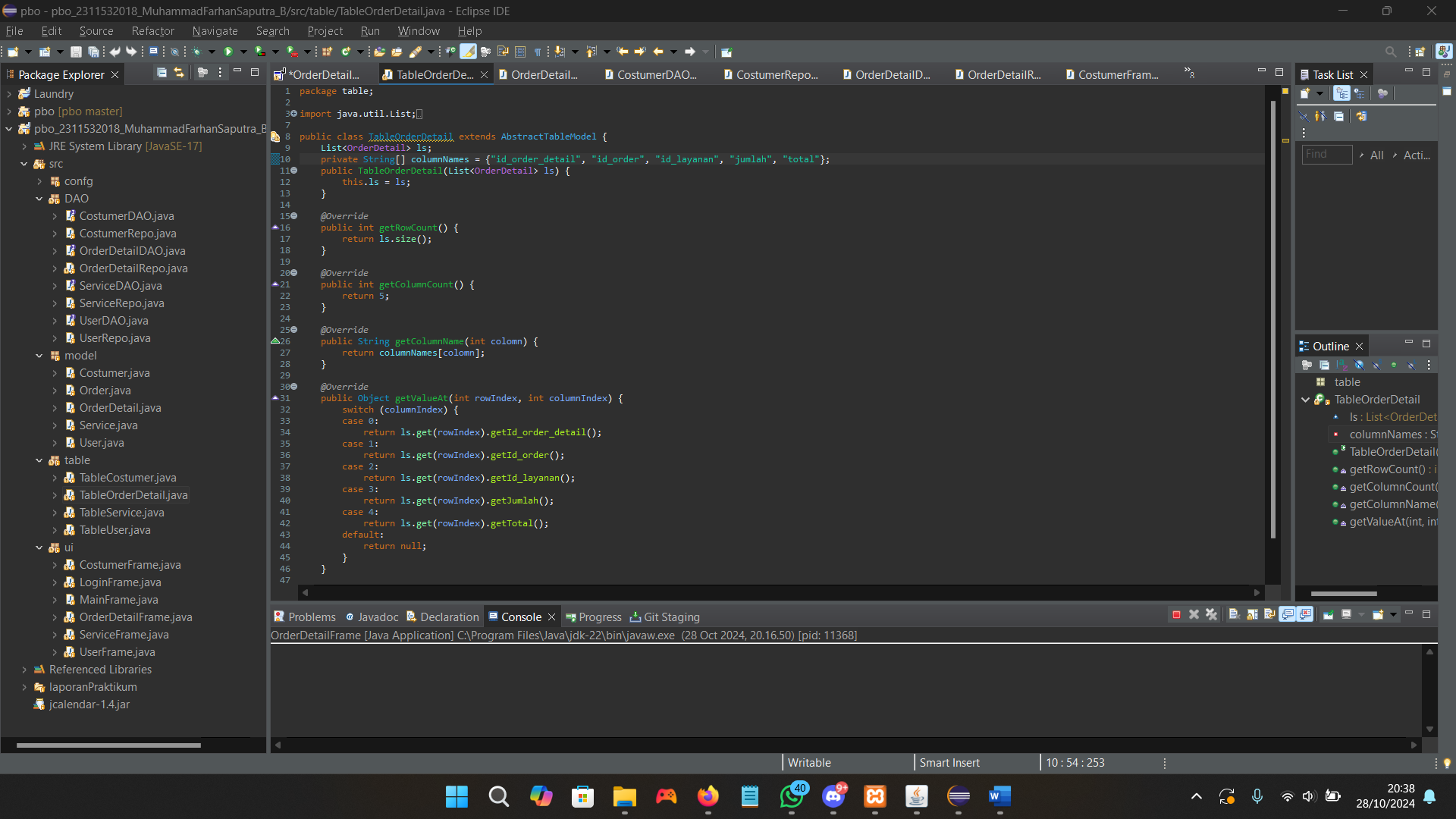
**Membuat OrderDetail**

1. **Membuat Model OrderDetail**



Pada OrderDetail akan dibuat 5 kolom, yaitu id\_order\_detail sebagai primary key dari table tersebut, id\_order merupakan foreign key dari table order yang akan dibuat pada praktikum selanjutnya, id\_layanan merupakan foreign key dari table service, jumlah merupakan jumlah service yang diambil, dan total merupakan hasil perkalian dari jumlah dan harga dari service.

1. **Membuat table OrderDetail di java**



Secara umum, class TabelOrderDetail akan melakukan beberapa proses, yaitu:

**1. Struktur Kelas**

* Kode ini merupakan bagian dari kelas yang bertanggung jawab untuk menampilkan detail pesanan dalam bentuk tabel. Kelas ini menggunakan daftar objek OrderDetail sebagai sumber data.

**2. Inisialisasi Data**

* Kelas ini menyimpan daftar OrderDetail yang diterima melalui konstruktor. Ini memungkinkan tabel untuk memiliki akses ke data yang akan ditampilkan.

**3. Jumlah Baris**

* Metode untuk menghitung jumlah baris dalam tabel mengembalikan ukuran dari daftar OrderDetail. Ini berarti jumlah baris yang ditampilkan akan sesuai dengan jumlah objek dalam daftar.

**4. Jumlah Kolom**

* Terdapat metode yang mengembalikan jumlah kolom tetap, yaitu lima kolom, sesuai dengan informasi yang ingin ditampilkan.

**5. Nama Kolom**

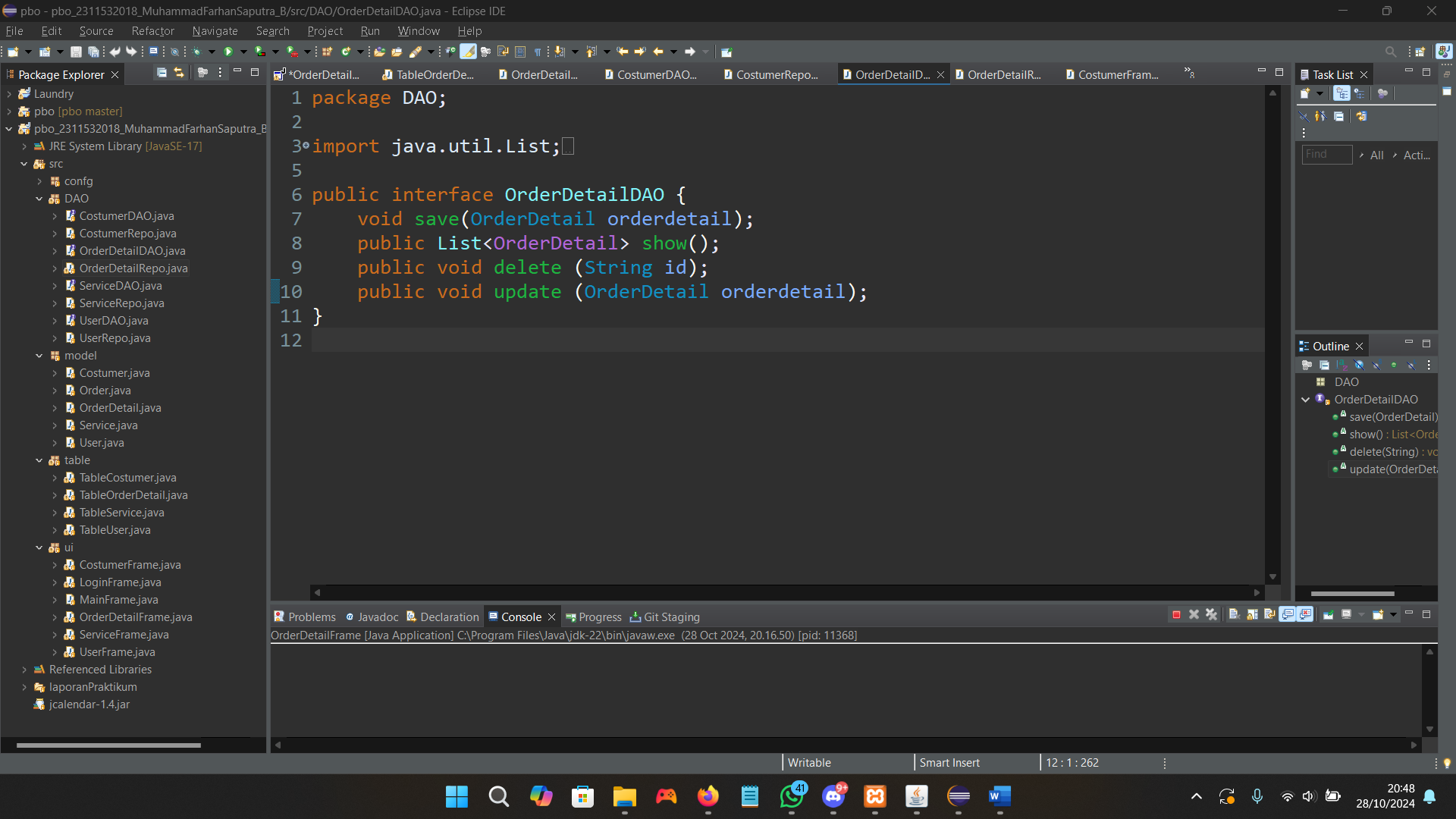
* Kelas ini memiliki array yang menyimpan nama-nama kolom. Metode yang ada memungkinkan untuk mendapatkan nama kolom berdasarkan indeks, sehingga tabel dapat menampilkan nama yang sesuai.

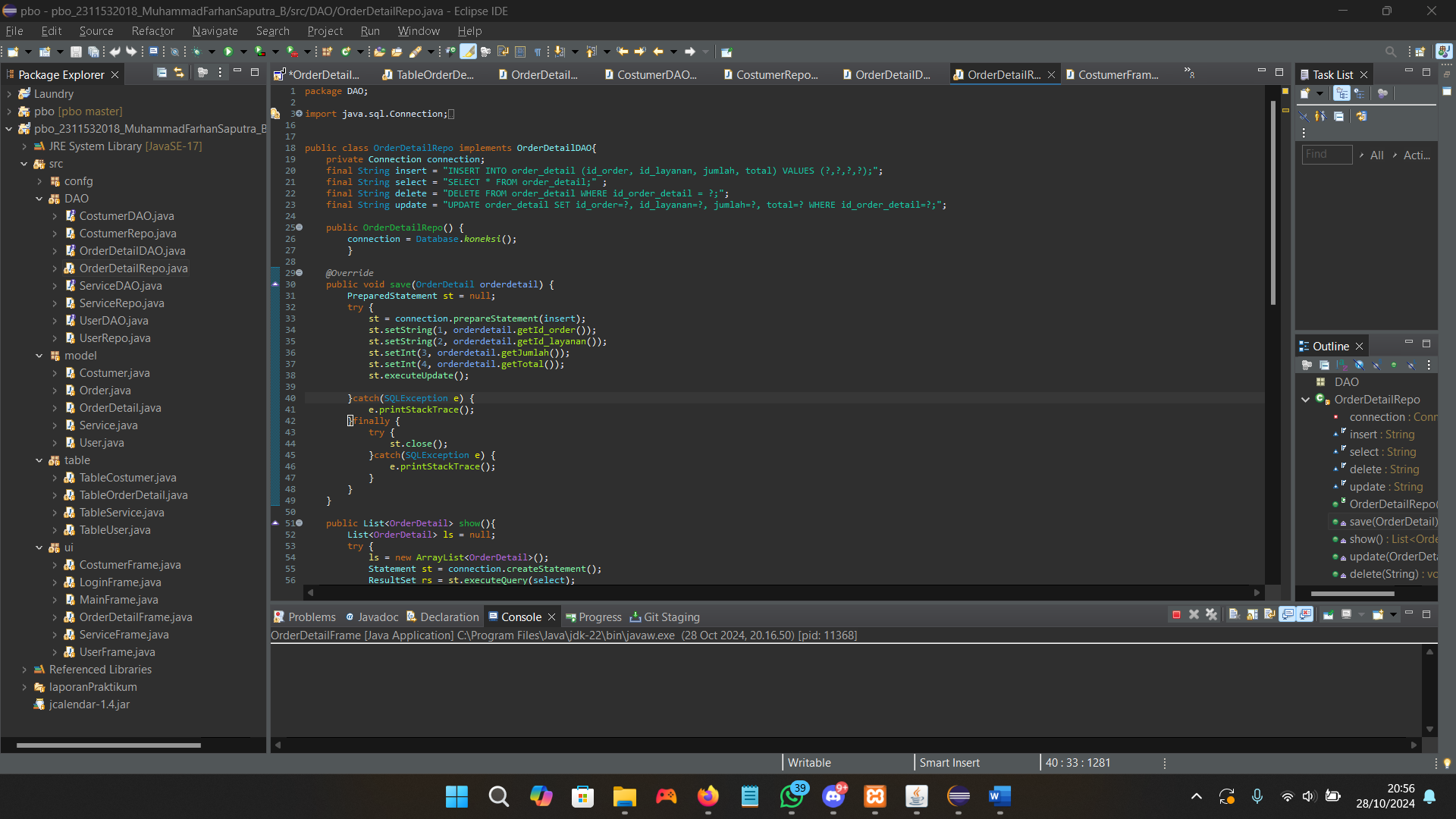
**6. Mengambil Data untuk Sel**

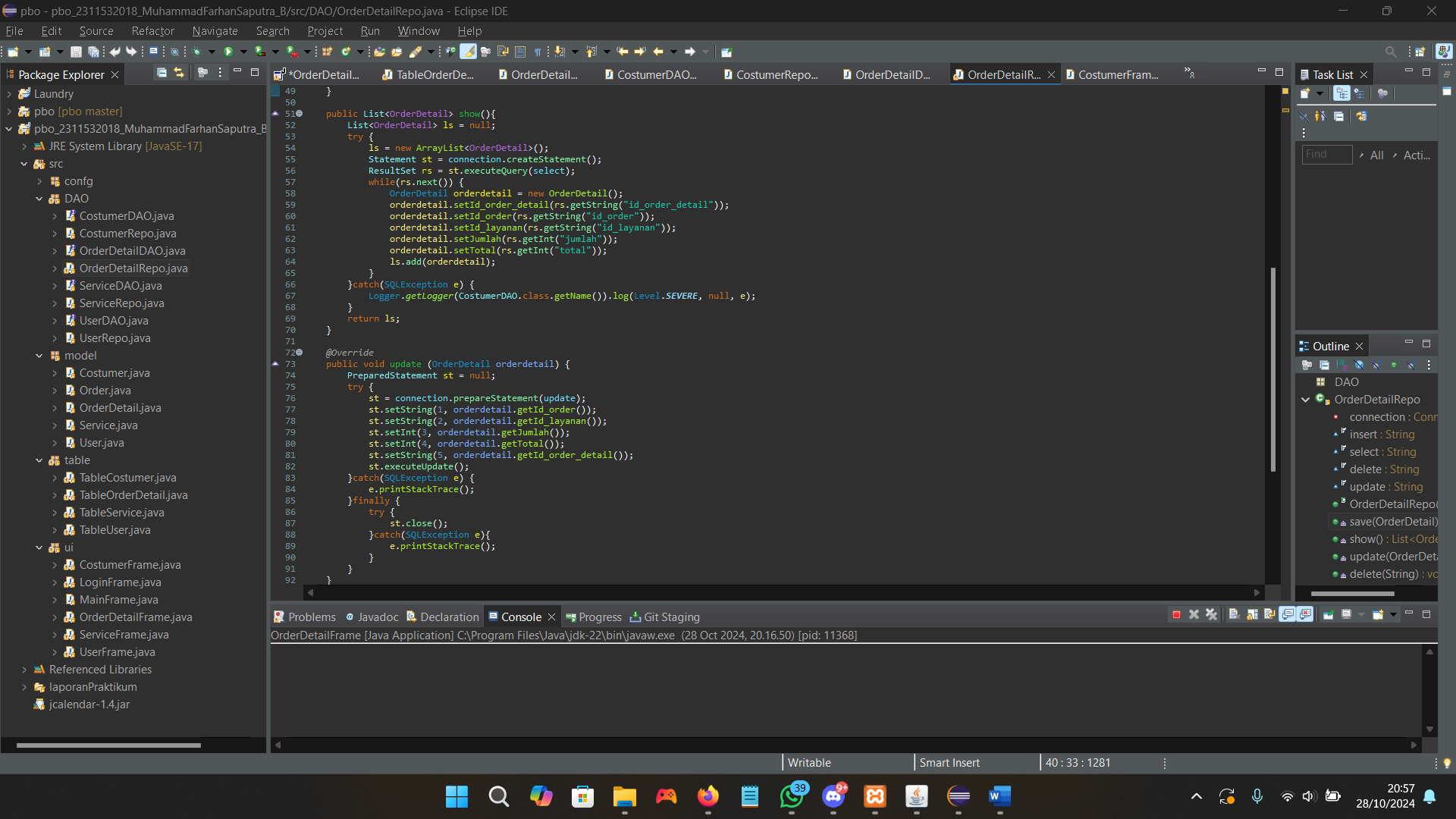
* Metode untuk mengambil nilai dari sel tertentu berdasarkan indeks baris dan kolom menggunakan struktur kontrol. Berdasarkan kolom yang diminta, metode ini mengakses objek OrderDetail yang sesuai dan mengembalikan nilai yang relevan, seperti ID pesanan, jumlah, dan total.

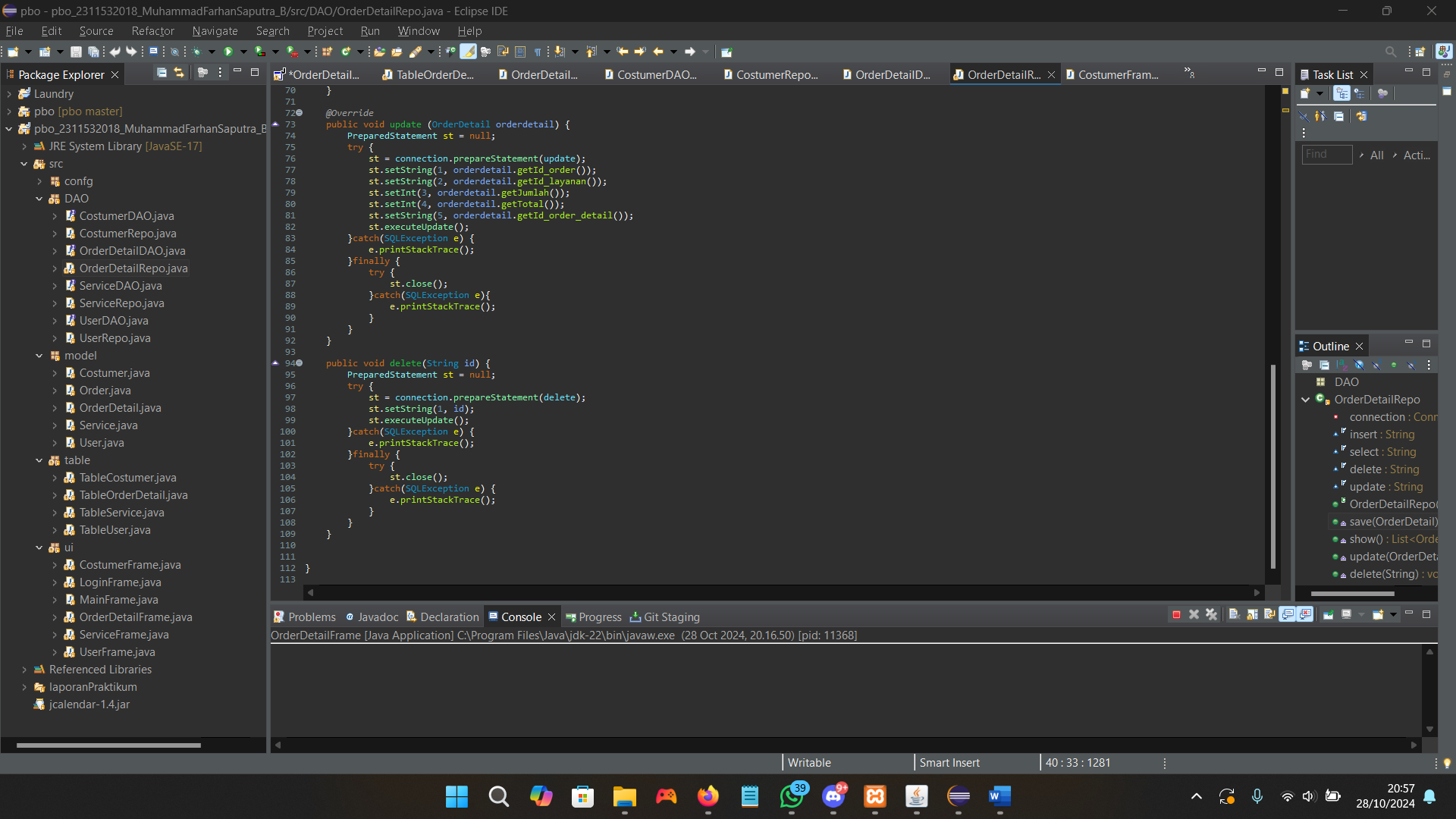
**Urutan Proses**

1. **Input Data**: Daftar objek OrderDetail dimasukkan ke dalam konstruktor kelas.
2. **Menghitung Baris dan Kolom**: Metode menghitung jumlah baris dan kolom yang akan ditampilkan.
3. **Menampilkan Nama Kolom**: Nama kolom diambil dari array dan ditampilkan di tabel.
4. **Mengambil Nilai**: Ketika sel tertentu diminta, nilai diambil dari objek OrderDetail sesuai dengan indeks yang diberikan.
5. **Membuat interface OrderDetailDAO dan class OrderDetailRepo**









**1. Interface: OrderDetailDAO**

* **Tujuan**: Interface ini mendefinisikan kontrak untuk operasi dasar yang dapat dilakukan pada objek OrderDetail. Ini termasuk metode untuk menyimpan, menampilkan, menghapus, dan memperbarui detail pesanan.
* **Metode**:
  + save(OrderDetail orderdetail): Menyimpan objek OrderDetail ke dalam basis data.
  + show(): Mengembalikan daftar semua objek OrderDetail yang ada di basis data.
  + delete(String id): Menghapus objek OrderDetail berdasarkan ID yang diberikan.
  + update(OrderDetail orderdetail): Memperbarui informasi dari objek OrderDetail yang sudah ada.

**2. Kelas: OrderDetailRepo**

* **Tujuan**: Kelas ini mengimplementasikan interface OrderDetailDAO dan menyediakan logika untuk berinteraksi dengan basis data menggunakan JDBC (Java Database Connectivity).

**3. Variabel dan Koneksi Database**

* **Koneksi**: Kelas ini memiliki variabel connection yang digunakan untuk berkomunikasi dengan basis data. Koneksi ini diinisialisasi dalam konstruktor dengan memanggil metode koneksi() dari kelas Database.
* **Query SQL**: Kelas ini menyimpan string untuk berbagai query SQL yang digunakan untuk operasi basis data, seperti INSERT, SELECT, DELETE, dan UPDATE.

**4. Metode save**

* **Fungsi**: Metode ini menyimpan objek OrderDetail ke dalam basis data.
* **Proses**:
  + Menggunakan PreparedStatement untuk menghindari SQL injection dan meningkatkan kinerja.
  + Mengatur parameter dari objek OrderDetail ke dalam query.
  + Menjalankan query untuk menyimpan data.
  + Menangani pengecualian SQL dan memastikan bahwa objek PreparedStatement ditutup setelah selesai.

**5. Metode show**

* **Fungsi**: Metode ini mengambil semua detail pesanan dari basis data dan mengembalikannya dalam bentuk daftar.
* **Proses**:
  + Membuat daftar baru untuk menyimpan objek OrderDetail.
  + Menggunakan Statement untuk mengeksekusi query SELECT.
  + Mengiterasi hasil query (ResultSet) dan mengisi objek OrderDetail dengan data yang diambil dari basis data.
  + Menangani pengecualian SQL dan mencatat kesalahan jika terjadi.

**6. Metode update**

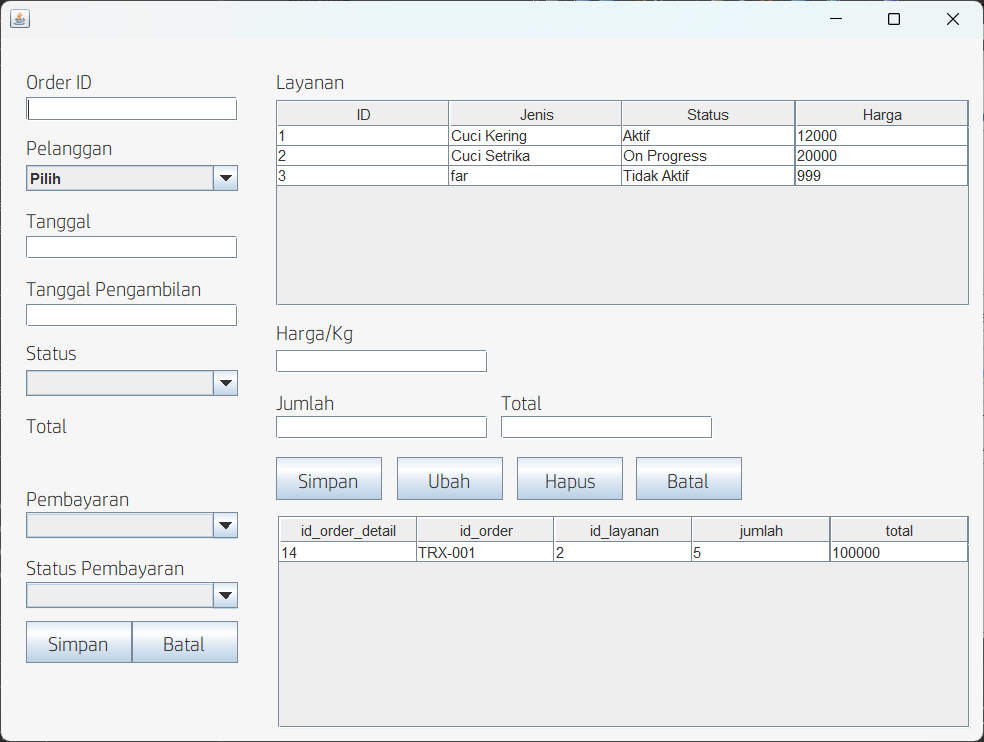
* **Fungsi**: Metode ini memperbarui informasi dari objek OrderDetail yang sudah ada di basis data.
* **Proses**:
  + Menggunakan PreparedStatement untuk menyiapkan dan menjalankan query UPDATE.
  + Mengatur parameter dari objek OrderDetail untuk memperbarui data yang sesuai.
  + Menangani pengecualian SQL dan memastikan bahwa objek PreparedStatement ditutup setelah selesai.

**Urutan Proses**

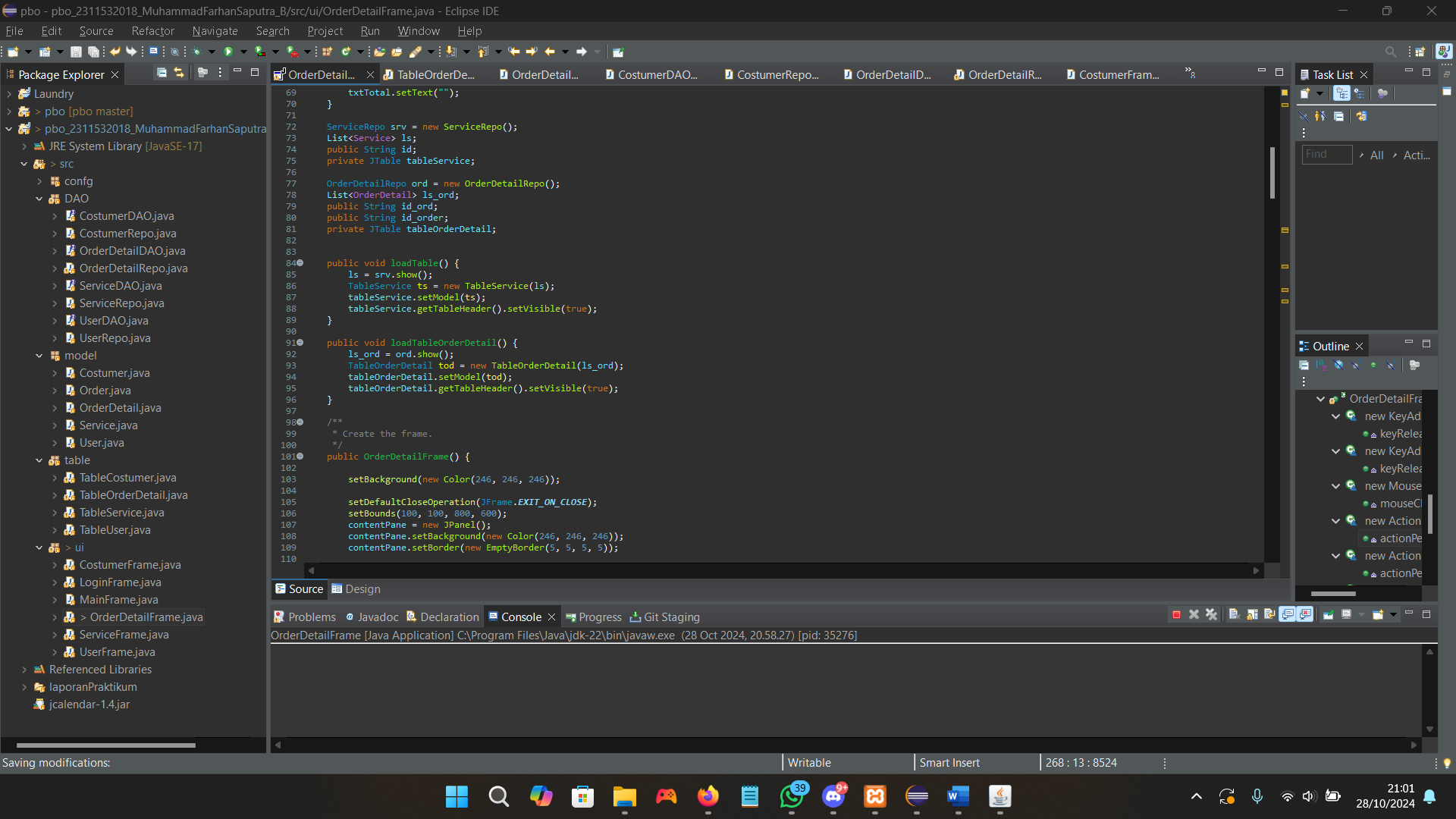
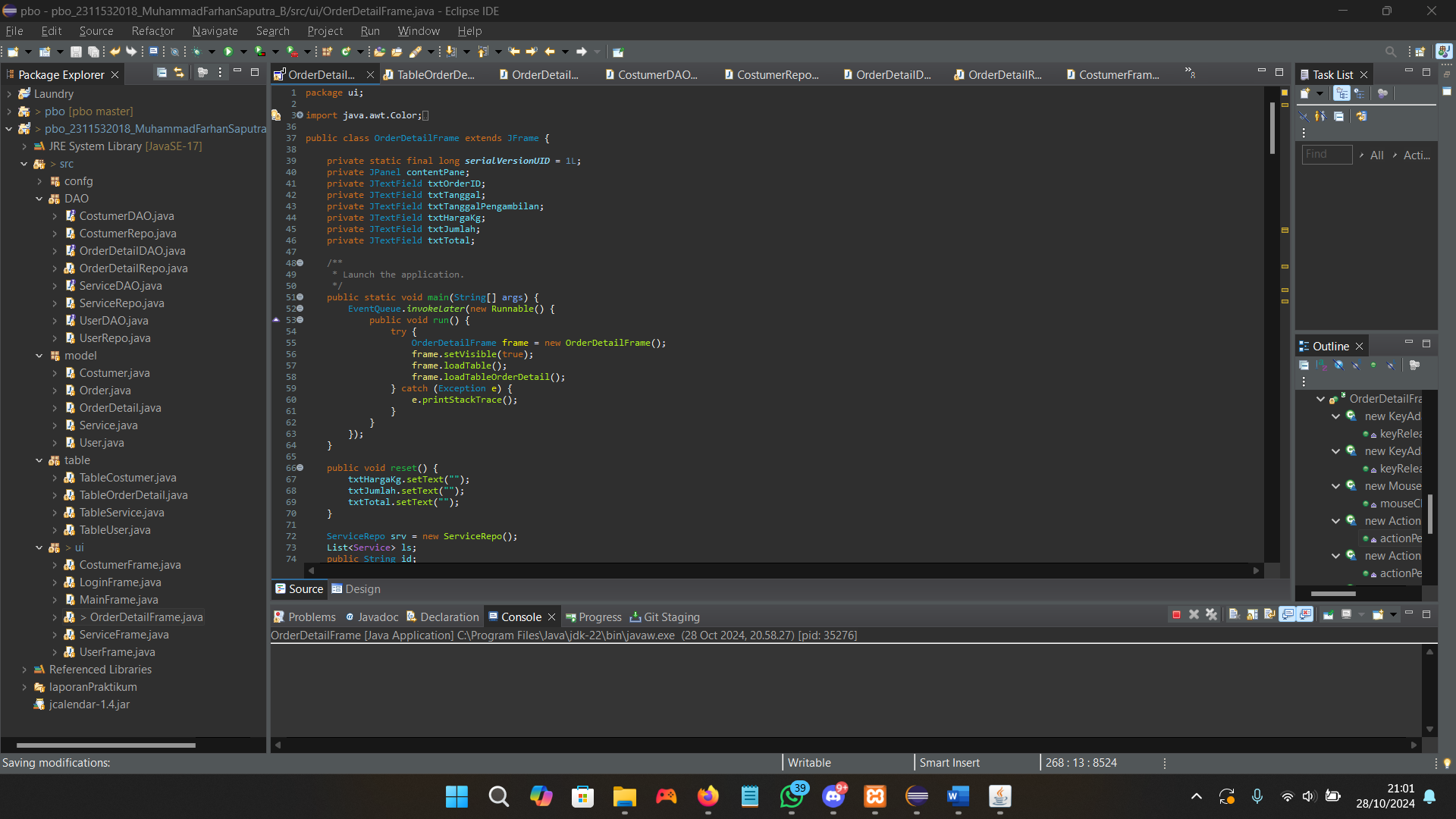
1. **Definisi Interface**: Interface OrderDetailDAO mendefinisikan metode yang diperlukan untuk mengelola objek OrderDetail.
2. **Implementasi Kelas**: Kelas OrderDetailRepo mengimplementasikan interface tersebut dan menyediakan logika untuk berinteraksi dengan basis data.
3. **Koneksi Database**: Koneksi ke basis data diatur dalam konstruktor kelas.
4. **Operasi CRUD**:
   * **Simpan**: Metode save digunakan untuk menambahkan detail pesanan baru.
   * **Tampilkan**: Metode show digunakan untuk mengambil semua detail pesanan yang ada.
   * **Perbarui**: Metode update digunakan untuk memperbarui detail pesanan yang sudah ada.

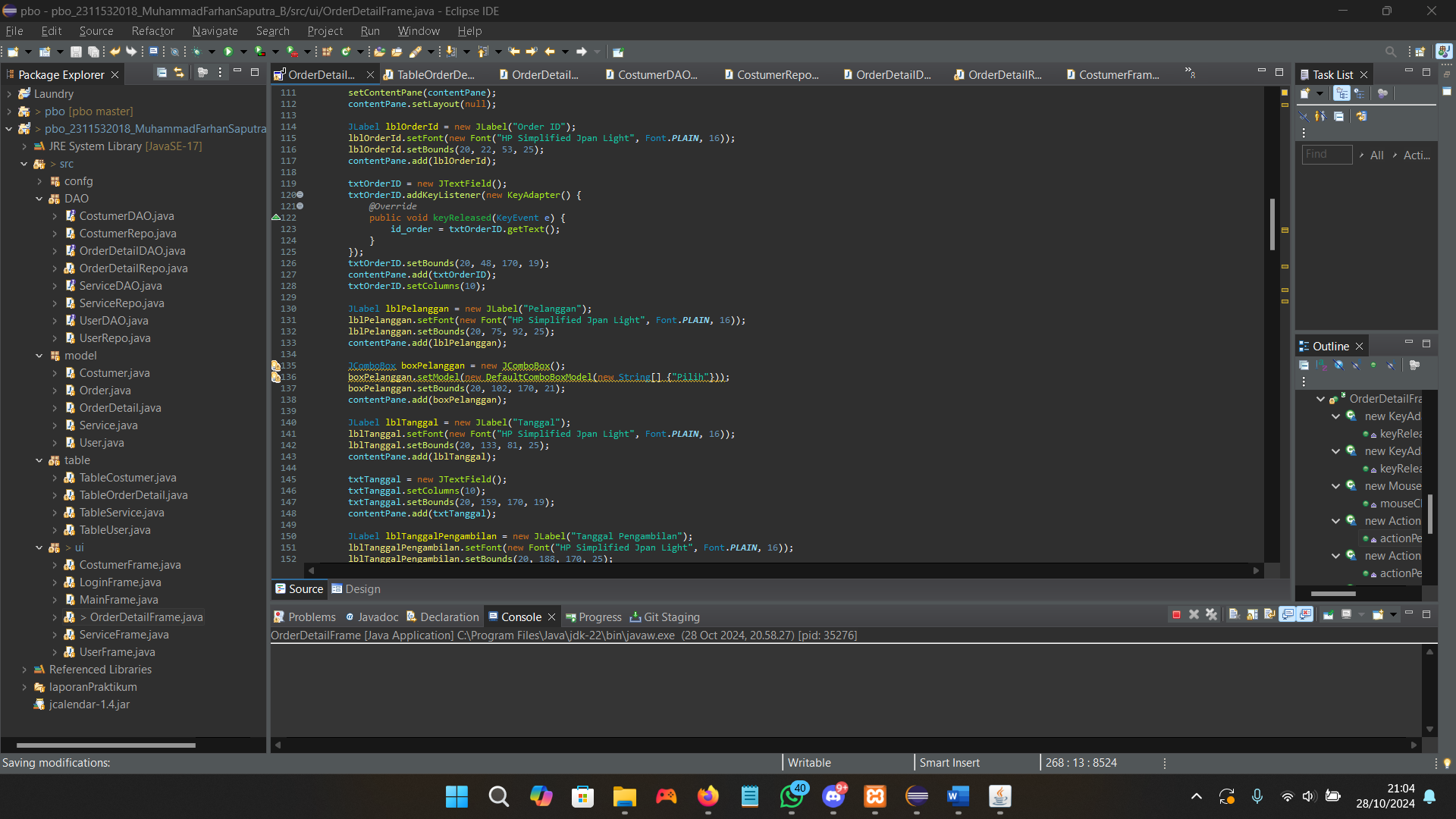
**4. Membuat OrderDetailFrame**

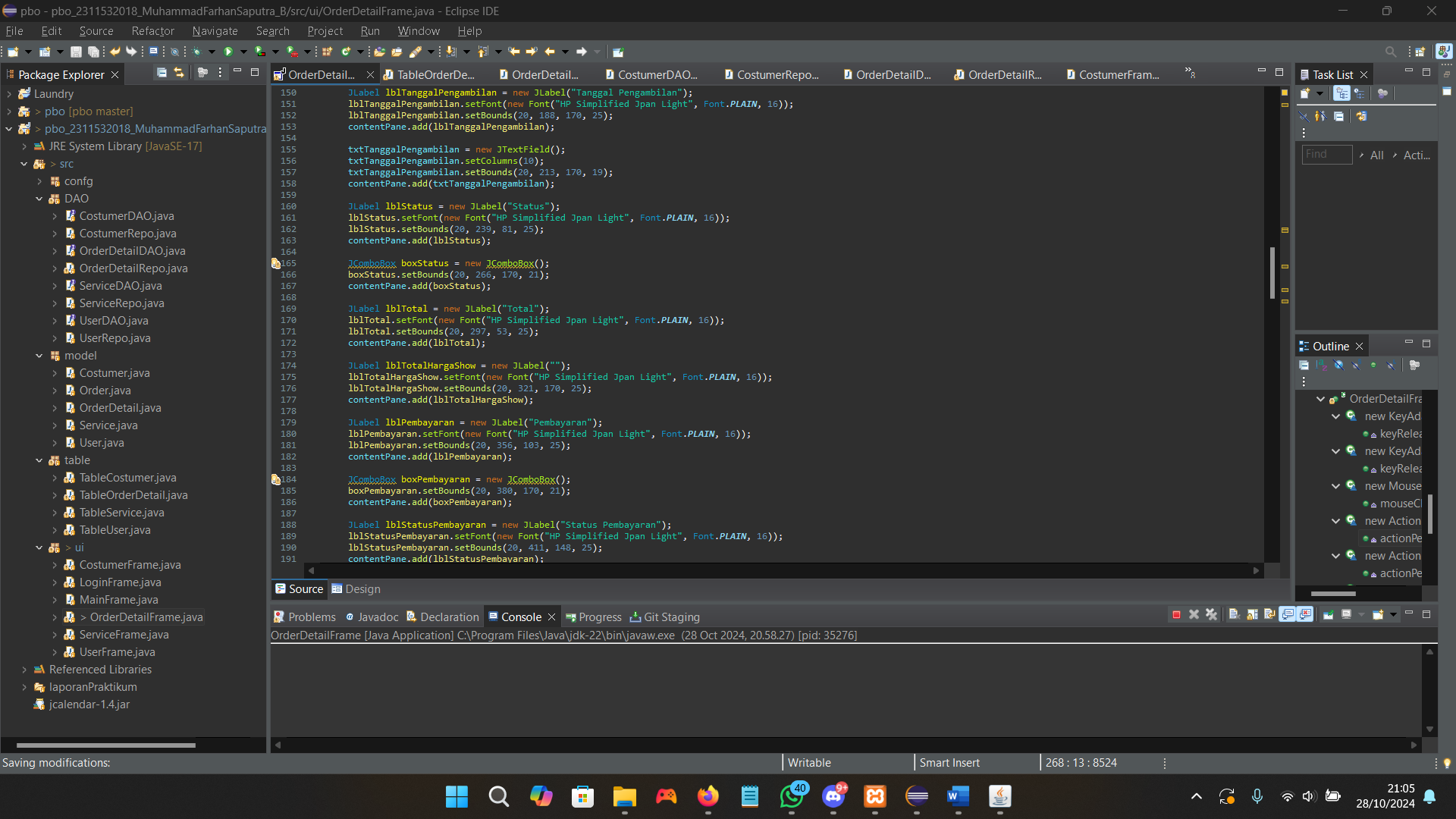
**a. Tampilan Desain OrderDetailFrame**

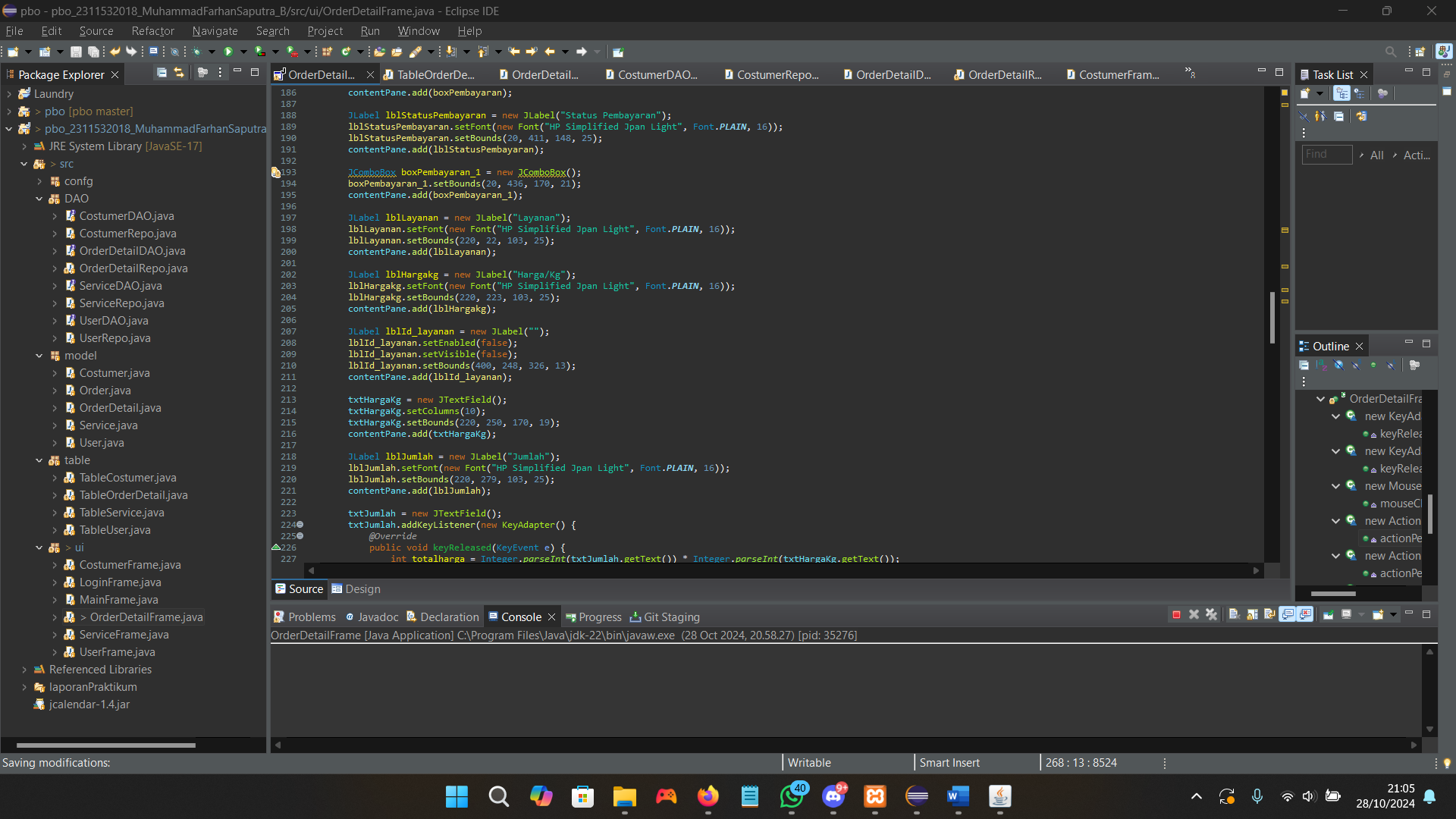
****

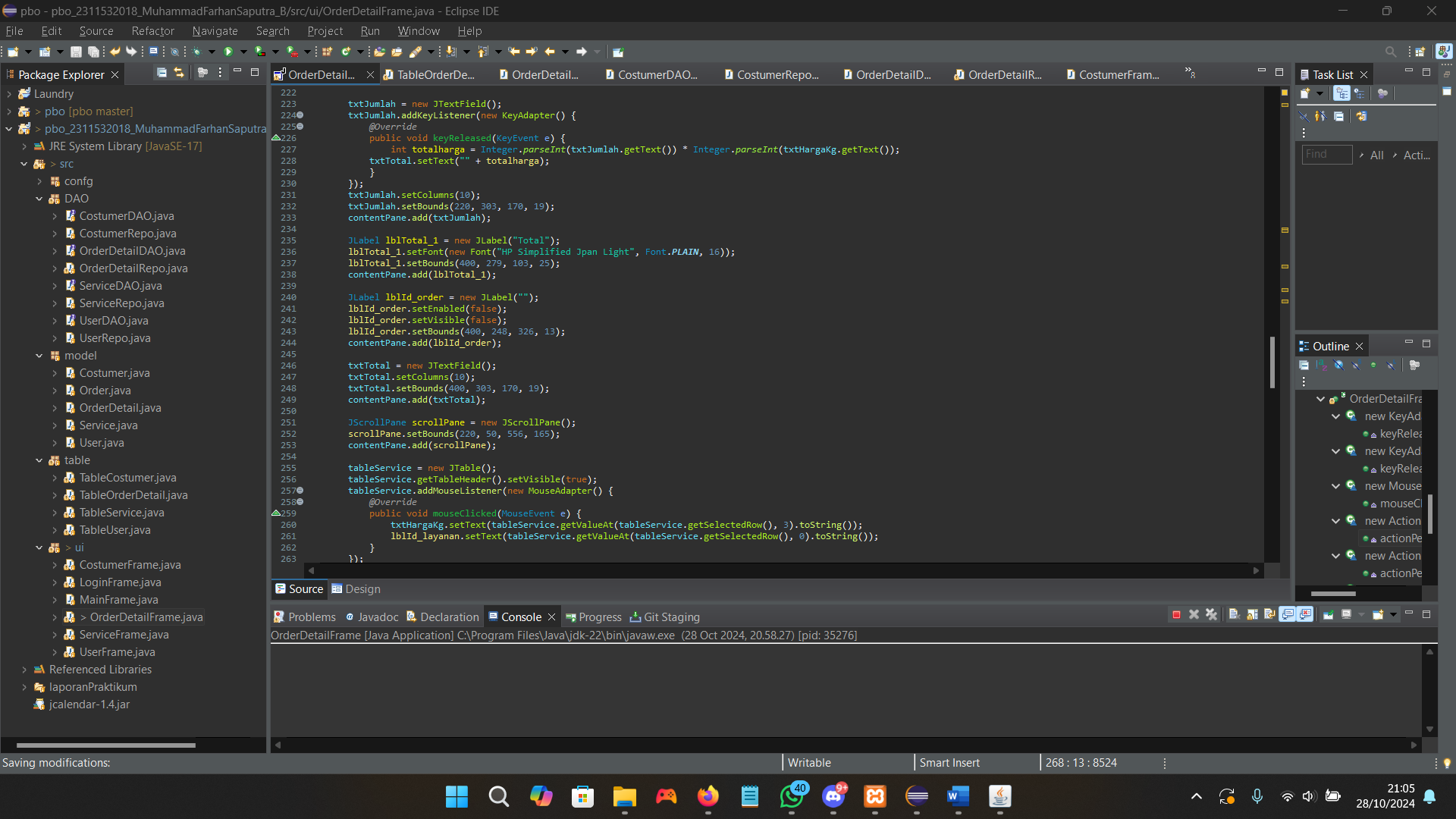
**b. Kode**

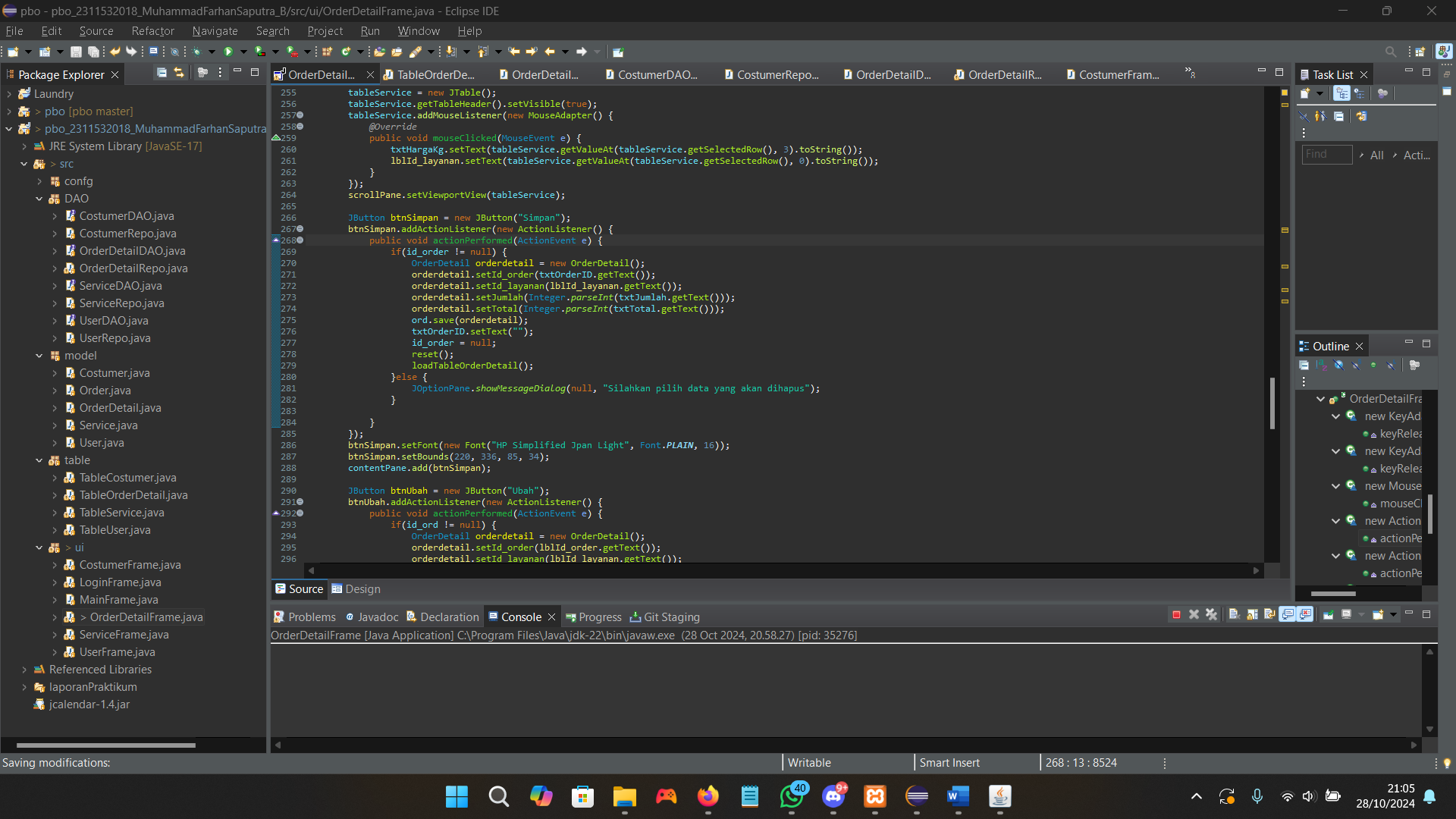


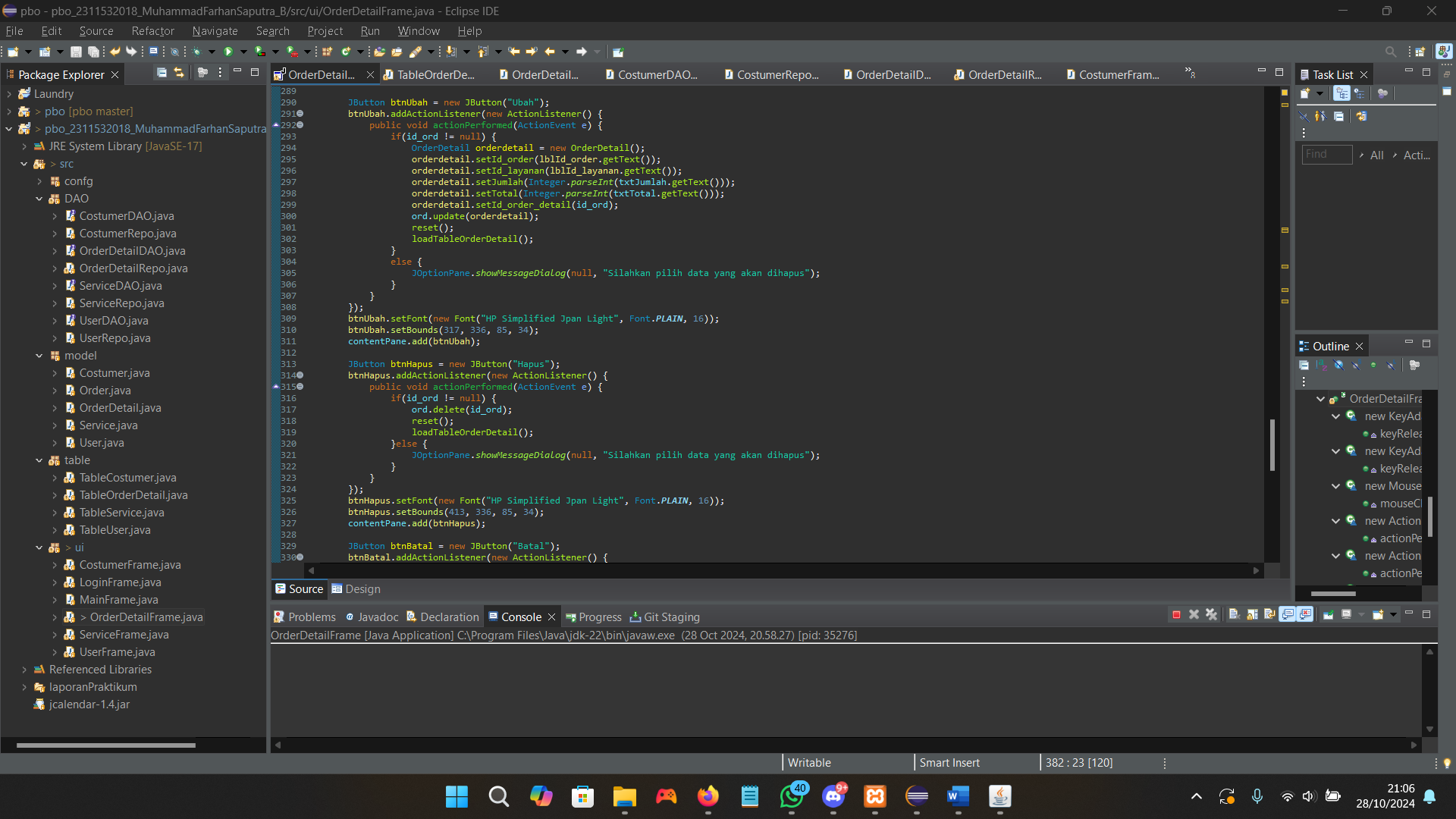


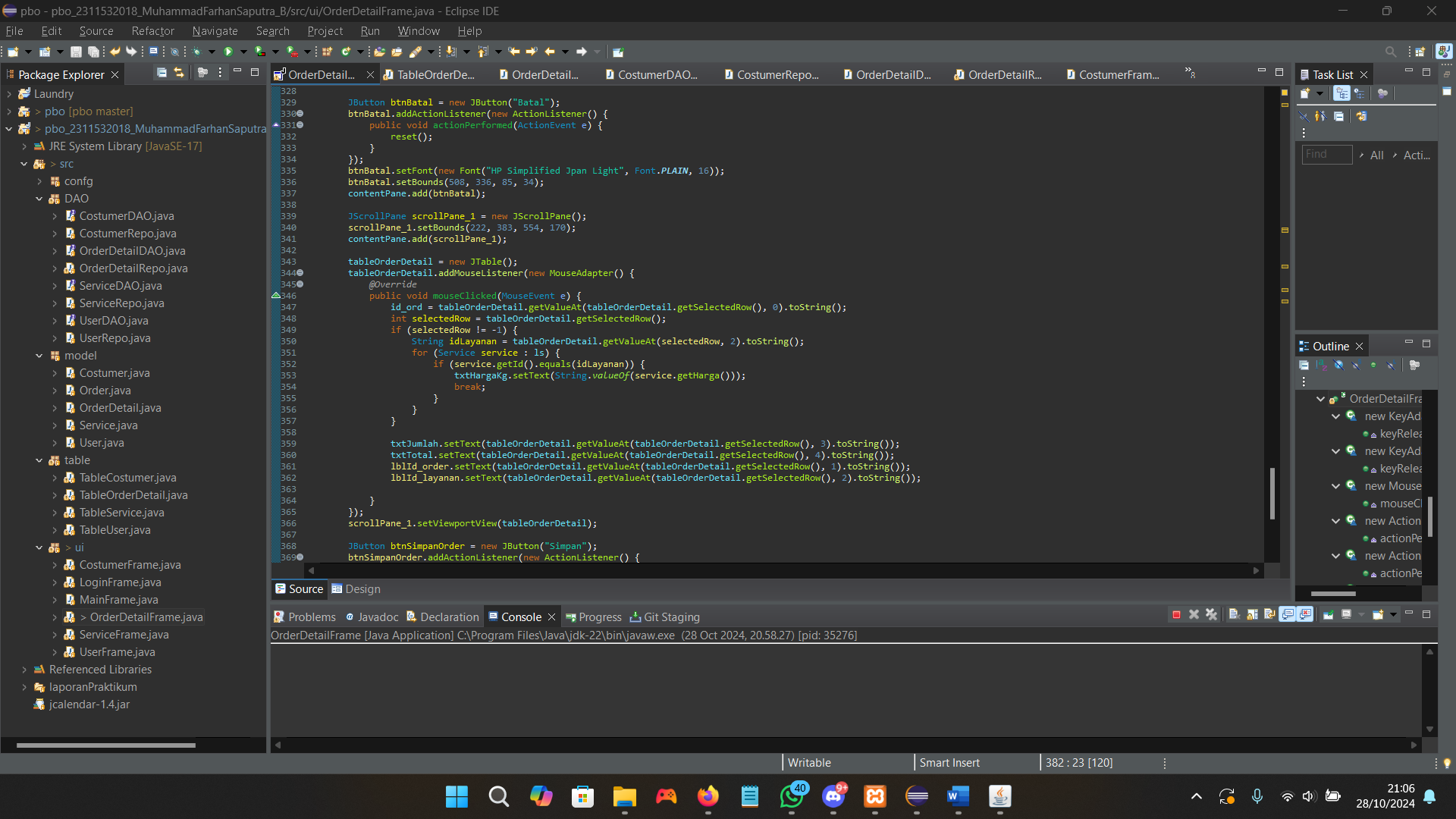


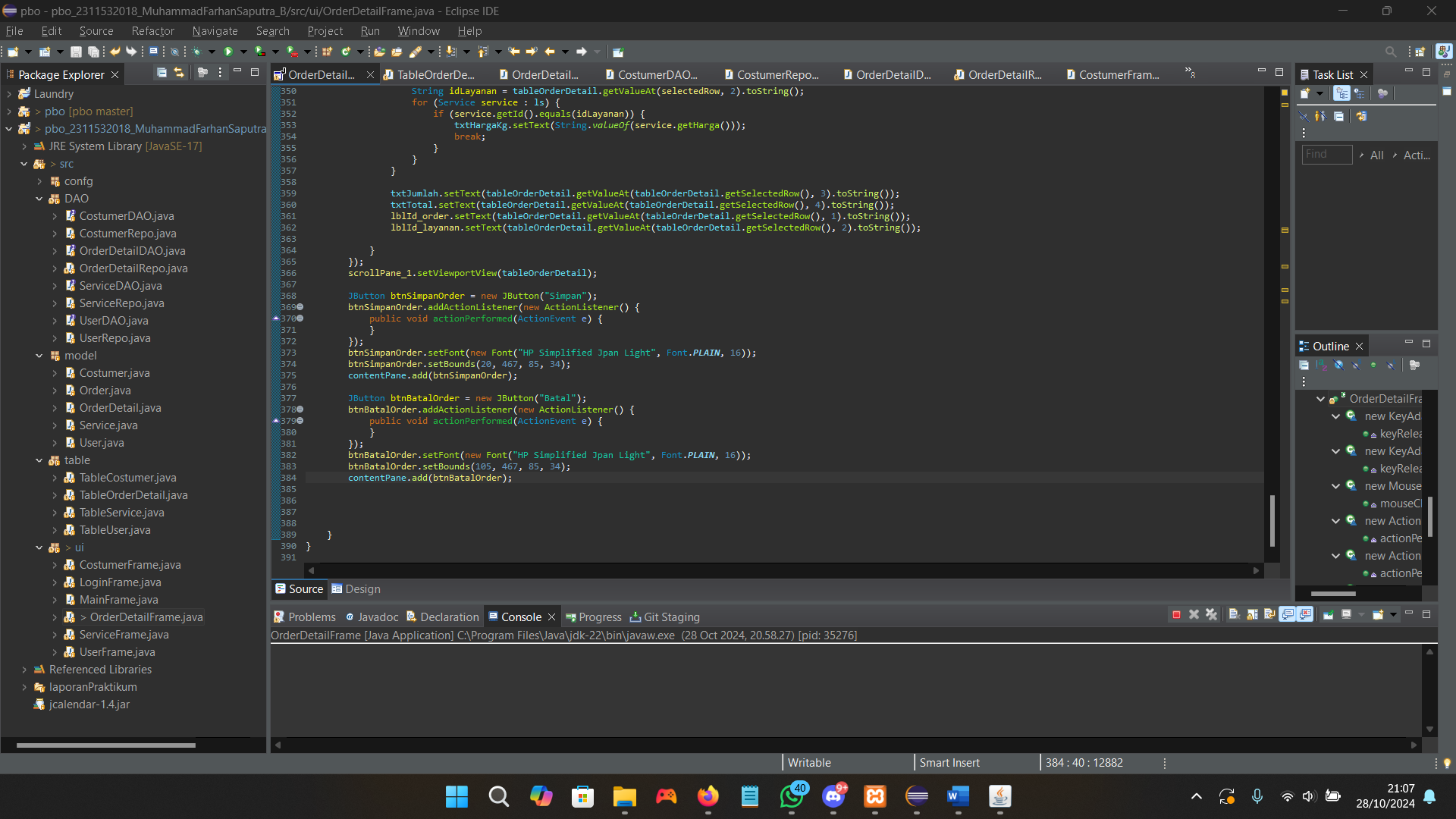












Ada beberapa hal penting yang terdapat pada kode tersebut:

**1. Import Tabel**

* **import table.TableOrderDetail;** dan **import table.TableService;**:
  + Keduanya mengimpor kelas yang bertanggung jawab untuk mengelola dan menampilkan data dalam bentuk tabel untuk objek OrderDetail dan Service. Kelas-kelas ini biasanya mengimplementasikan TableModel dari Swing, sehingga dapat digunakan untuk menampilkan data dalam komponen GUI seperti JTable.

**2. Metode loadTable**

* **Fungsi**: Metode ini digunakan untuk memuat dan menampilkan data layanan (Service) ke dalam tabel tableService.
* **Proses**:
  + Memanggil metode show() dari ServiceRepo, yang mengambil daftar semua layanan dari basis data.
  + Membuat objek TableService dengan daftar layanan yang diambil.
  + Mengatur model tabel tableService dengan objek TableService, sehingga data layanan dapat ditampilkan dalam tabel.
  + Menampilkan header tabel dengan memanggil getTableHeader().setVisible(true).

**3. Metode loadTableOrderDetail**

* **Fungsi**: Metode ini digunakan untuk memuat dan menampilkan data detail pesanan (OrderDetail) ke dalam tabel tableOrderDetail.
* **Proses**:
  + Memanggil metode show() dari OrderDetailRepo, yang mengambil daftar semua detail pesanan dari basis data.
  + Membuat objek TableOrderDetail dengan daftar detail pesanan yang diambil.
  + Mengatur model tabel tableOrderDetail dengan objek TableOrderDetail, sehingga data detail pesanan dapat ditampilkan dalam tabel.
  + Menampilkan header tabel dengan memanggil getTableHeader().setVisible(true).

**4. Tombol Simpan**

* **Fungsi**: Tombol ini digunakan untuk menyimpan detail pesanan baru ke dalam basis data.
* **Proses**:
  + Menggunakan ActionListener untuk mendefinisikan tindakan yang diambil saat tombol diklik.
  + Memeriksa apakah id\_order tidak null (artinya ada order yang dipilih).
  + Jika ada, membuat objek OrderDetail, mengatur atributnya berdasarkan input dari pengguna (seperti ID order, ID layanan, jumlah, dan total).
  + Memanggil metode save dari OrderDetailRepo untuk menyimpan objek ke dalam basis data.
  + Mengosongkan input dan memanggil metode reset() untuk mengatur ulang form, serta memuat ulang tabel detail pesanan.

**5. Tombol Ubah**

* **Fungsi**: Tombol ini digunakan untuk memperbarui detail pesanan yang sudah ada.
* **Proses**:
  + Menggunakan ActionListener untuk mendefinisikan tindakan saat tombol diklik.
  + Memeriksa apakah id\_ord tidak null (artinya ada detail pesanan yang dipilih).
  + Jika ada, membuat objek OrderDetail, mengatur atributnya, dan menyertakan id\_order\_detail untuk memperbarui entri yang tepat di basis data.
  + Memanggil metode update dari OrderDetailRepo untuk memperbarui objek di basis data.
  + Mengosongkan input dan memanggil metode reset() untuk mengatur ulang form, serta memuat ulang tabel detail pesanan.

**6. Tombol Hapus**

* **Fungsi**: Tombol ini digunakan untuk menghapus detail pesanan yang dipilih dari basis data.
* **Proses**:
  + Menggunakan ActionListener untuk mendefinisikan tindakan saat tombol diklik.
  + Memeriksa apakah id\_ord tidak null (artinya ada detail pesanan yang dipilih).
  + Jika ada, memanggil metode delete dari OrderDetailRepo untuk menghapus entri yang tepat di basis data.
  + Memanggil metode reset() untuk mengatur ulang form dan memuat ulang tabel detail pesanan.

**7. Tombol Batal**

* **Fungsi**: Tombol ini digunakan untuk mengatur ulang form input.
* **Proses**:
  + Menggunakan ActionListener untuk mendefinisikan tindakan saat tombol diklik.
  + Memanggil metode reset() untuk mengosongkan semua field input yang ada.

**8. JTextField dengan Metode**

* **txtJumlah** dan **txtHargaKg**:
  + Keduanya memiliki KeyListener untuk menangani peristiwa ketika pengguna mengetikkan nilai