Laporan UAS Prak PBO

IMPLEMENTASI KONSEP OBJECT-ORIENTED PROGRAMMING PADA PROJECT UAS

Disusun untuk memenuhi tugas Mata Kuliah
Praktikum Pemrograman Berorientasi Objek

oleh:

Cut Mutia Rahmah	(2408107010062)
Teuku Fikram Al Syahbanna	(2408107010044)
Ahmad Daniel Chalid	(2408107010061)
Inayah Kamila Nurman	(2408107010060)

JURUSAN INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

2025

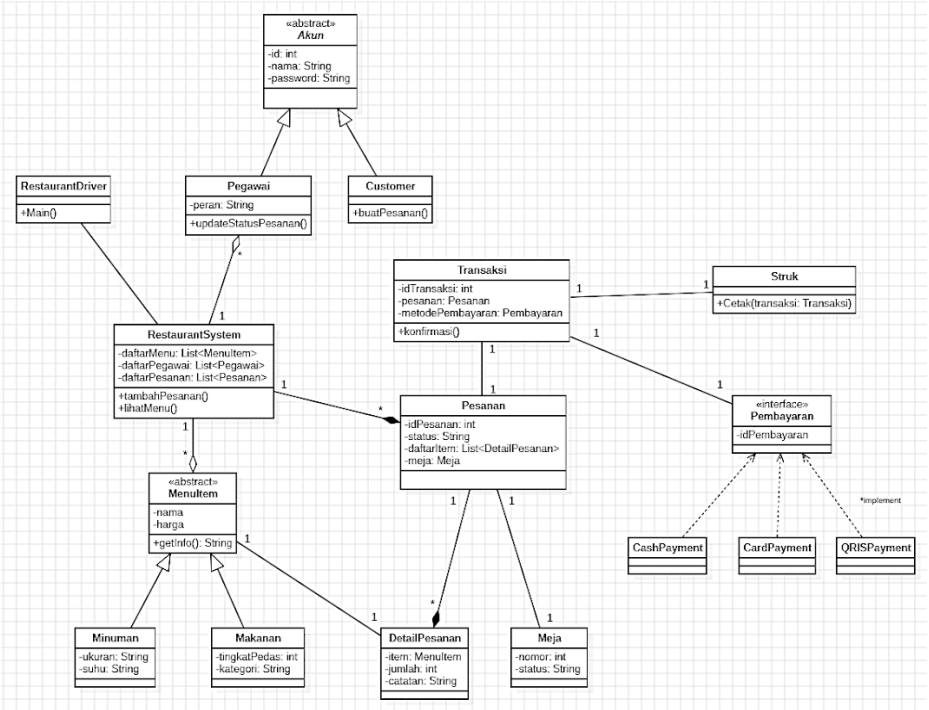
A. Pendahuluan

Laporan ini disusun untuk menjelaskan bagaimana konsep Object-Oriented Programming diterapkan pada project UAS mata kuliah Pemrograman Berorientasi Objek. Project yang dibuat merupakan sistem pemesanan restoran sederhana yang terdiri dari beberapa komponen utama, yaitu model, service, GUI, dan class utama untuk menjalankan program. Setiap komponen dirancang menggunakan prinsip dasar OOP seperti encapsulation, inheritance, polymorphism, abstraction, dan composition agar sistem lebih terstruktur, modular, dan mudah dikembangkan.

Pada bagian model, setiap class disusun untuk merepresentasikan objek nyata seperti akun, pelanggan, pegawai, menu, pesanan, transaksi, hingga struk pembayaran. Setiap class dilengkapi validasi dan mekanisme error handling untuk memastikan setiap objek selalu berada dalam kondisi yang benar. Bagian service berfungsi menangani logika bisnis, seperti proses pembayaran dan pengelolaan alur sistem. Sementara itu, GUI dirancang untuk memberikan tampilan interaktif kepada pengguna, dan file AppLauncher serta Main digunakan sebagai titik masuk untuk menjalankan aplikasi.

Melalui laporan ini, setiap file akan dijelaskan fungsinya dan dipetakan dengan konsep OOP yang relevan berdasarkan implementasi yang telah dibuat pada project. Dengan pendekatan ini, pembaca dapat memahami bagaimana teori OOP diimplementasikan dalam bentuk program yang utuh dan berjalan secara sistematis.

class diagram_pbo:



B. Implementasi pada source

2.1 GUI

Folder GUI berisi seluruh tampilan berbasis antarmuka grafis yang digunakan pengguna untuk berinteraksi secara visual dengan sistem restoran. Tidak seperti Main.java yang berjalan pada mode console, folder GUI menyediakan versi modern dari aplikasi dengan tombol, form input, window navigasi, dan tampilan yang lebih mudah digunakan. Setiap file GUI mewakili satu fungsi atau satu peran dalam sistem, seperti layar login, dashboard pegawai, halaman pemesanan, hingga menu pembayaran.

Komponen GUI dalam folder ini bekerja dengan memanfaatkan event listener, aksi tombol, serta interaksi dengan object dari RestaurantSystem dan class model untuk menampilkan data dan melakukan proses seperti pemesanan, manajemen menu, atau pembayaran. Dengan menggunakan GUI, sistem menjadi lebih interaktif dan pengguna dapat mengoperasikan aplikasi tanpa mengetik perintah secara manual.

File yang terdapat pada GUI terdiri dari:

a. CustomerGUI.java
b. DashboardGUI.java
c. KasirGUI.java
d. KokiGUI.java
e. LoginGUI.java
f. MenuGUI.java
g. OrderGUI.java
h. PaymentGUI.java
i. PelayanGUI.java

Penerapan Konsep OOP

Folder GUI menggunakan berbagai konsep OOP untuk memisahkan logika program dari tampilan, menjaga keteraturan kode, dan membuat aplikasi mudah dikembangkan. Konsep yang dominan adalah pemisahan antara tampilan antarmuka pengguna dengan data dan proses bisnis. Setiap tampilan seperti login, dashboard, kasir, koki, pelayan, hingga menu dan order dibuat dalam class khusus sehingga perannya terkontrol dengan baik dan tidak bercampur dengan logika inti.

Setiap class GUI memperlihatkan penggunaan enkapsulasi melalui komponen dan variabel private yang hanya bisa diakses lewat method internal, sehingga setiap jendela atau panel dapat menjaga data UI-nya tanpa gangguan dari luar.

Selain itu terlihat juga penerapan abstraksi yang membuat setiap GUI hanya fokus pada hal visual dan interaksi pengguna, sedangkan proses bisnis seperti menambah menu, memproses pembayaran, atau mengatur status meja sepenuhnya dikerjakan oleh RestaurantSystem yang berada di luar folder GUI.

Konsep pemanggilan objek dan penggunaan objek lintas class sangat kuat dalam folder ini. Hampir seluruh GUI menerima objek RestaurantSystem sebagai parameter konstruktor. Ini menunjukkan pemanfaatan object collaboration di mana setiap bagian antarmuka membutuhkan data atau layanan dari objek yang lebih besar. Ketika kasir membuka PaymentGUI atau pelayan membuka OrderGUI, mereka tidak membuat logika baru tetapi menggunakan objek-objek seperti Pesanan, Pegawai, Meja, dan Pembayaran yang berasal dari model dan service.

Pada beberapa bagian, polymorphism juga digunakan, terutama ketika class-class GUI memanggil metode prosesPembayaran melalui interface Pembayaran. GUI tidak perlu tahu apakah yang diproses itu CashPayment, CardPayment, atau QRISPayment. Mereka hanya memanggil metode yang sama dan membiarkan class yang sesuai menjalankan logikanya. Hal ini membuat kode GUI menjadi lebih fleksibel terhadap perubahan jenis pembayaran baru di masa depan.

Desain GUI juga menerapkan pemisahan tanggung jawab. DashboardGUI berfungsi sebagai pusat navigasi, sedangkan class lain hanya mengurus fungsi spesifik seperti menampilkan menu, membuat pesanan, atau melunasi pembayaran. Setiap tampilan diarahkan untuk hanya mengerjakan apa yang menjadi tugasnya sehingga struktur aplikasinya mudah dipahami dan dikembangkan.

Keseluruhan folder GUI memperlihatkan penerapan kombinasi konsep dasar OOP seperti abstraksi, enkapsulasi, dan polymorphism, serta prinsip pemisahan tugas yang membuat aplikasi restoran menjadi modular dan mudah diperluas.

2.2 Model

a. Akun.java

Class Akun merupakan class abstrak yang menjadi *kelas induk* untuk semua jenis akun dalam sistem, seperti Customer dan Pegawai. Class ini menyimpan data dasar seorang pengguna, yaitu id, nama, dan password, dan memastikan seluruh turunan memiliki atribut tersebut.

Class ini tidak dapat dibuat objek langsung, karena hanya berfungsi sebagai template umum untuk jenis akun lainnya.

Penerapan Konsep OOP

1. Encapsulation

   Class ini menggunakan enkapsulasi dengan menyimpan seluruh atribut dalam bentuk private. Semua akses dan perubahan nilai dilakukan melalui getter dan setter, sehingga data sensitif seperti password tetap terkontrol. Selain itu, setter dilengkapi validasi untuk memastikan data tidak kosong atau tidak valid.
2. Inheritance

   Akun merupakan class induk yang dirancang untuk diwariskan ke class lain. Customer, Pegawai, dan jenis akun lain mewarisi struktur dasar ini sehingga tidak perlu mendefinisikan ulang atribut umum. Hal ini membuat kode lebih rapi dan menghindari duplikasi.
3. Polymorphism

   Konsep polymorphism terlihat dari pemanfaatan method toString yang dapat dioverride oleh class turunan. Class Akun menyediakan bentuk dasar, tetapi subclass bebas memberikan representasi output yang berbeda sesuai kebutuhan masing-masing.
4. Abstraction

   Akun dibuat sebagai class abstrak untuk memberikan gambaran umum tentang apa yang harus dimiliki sebuah akun. Detail perilaku spesifik tidak ditentukan di sini, sehingga class ini menyembunyikan kompleksitas dan membiarkan class turunan menentukan implementasinya.
5. Error Handling

   Constructor dan setter menerapkan validasi yang melempar IllegalArgumentException ketika nilai id tidak valid atau ketika nama dan password kosong. Hal ini membuat object Akun hanya dapat dibuat ketika data benar, sehingga menjaga integritas sistem.

b. Customer.java

Customer adalah class turunan dari Akun yang mewakili pengguna sistem yang berperan sebagai pelanggan restoran. Class ini menyimpan data identitas pelanggan dari class Akun dan menambahkan fitur khusus berupa daftar riwayat pesanan yang pernah dibuat oleh customer. Setiap riwayat pesanan disimpan dalam bentuk id pesanan.

Penerapan Konsep OOP

1. Inheritance

   Customer mewarisi atribut dan method dari class Akun. Karena Akun merupakan class abstrak, Customer harus mengimplementasikan struktur yang sudah diberikan Akun, termasuk id, nama, dan password.

Dengan pewarisan ini, Customer tidak perlu mendefinisikan kembali atribut dasar yang sudah tersedia.

2. Encapsulation

   Atribut riwayatPesanan dibuat private agar data hanya bisa diakses dan dimodifikasi melalui method yang disediakan, yaitu getRiwayatPesanan dan tambahRiwayatPesanan. Dengan enkapsulasi ini, data transaksi pelanggan terhindar dari perubahan langsung oleh class luar.

3. Composition

   Customer memiliki sebuah list yang berisi id pesanan. Relasi ini menggambarkan bahwa customer dapat memiliki banyak pesanan, dan daftar riwayat tersebut hanya milik object customer. Jika object customer dihapus, riwayat pesanan miliknya juga ikut hilang sehingga relasi ini termasuk composition.

4. Error Handling

   Method tambahRiwayatPesanan menerapkan validasi input untuk menjaga konsistensi data. Validasi meliputi pengecekan apakah id pesanan lebih dari 0 dan apakah id tersebut sudah ada sebelumnya dalam list. Jika input tidak valid, sistem melempar IllegalArgumentException. Validasi ini mencegah duplikasi dan data pesanan yang tidak logis.

5. Polymorphism

   Method toString dioverride untuk menampilkan informasi customer dalam bentuk yang lebih spesifik. Class turunan memberikan perilaku yang berbeda dari class induknya, sehingga menjadi contoh penerapan polymorphism melalui overriding.

c. DetailPesanan.java

Class DetailPesanan digunakan untuk merepresentasikan satu item pesanan dalam sebuah transaksi. Setiap detail berisi menu yang dipesan, jumlah pesanan, dan catatan tambahan dari pelanggan. Class ini menjadi bagian penting dalam pembentukan pesanan karena memungkinkan satu pesanan memiliki banyak item berbeda beserta informasinya.

Penerapan Konsep OOP

1. Encapsulation

   Class ini menerapkan enkapsulasi melalui atribut private yang hanya bisa diakses lewat getter. Dengan cara ini, data seperti jumlah pesanan dan catatan menjadi lebih terkontrol dan tidak dapat diubah sembarangan dari luar. Konstruktor mengatur nilai awal dan memastikan tidak ada data yang tidak valid masuk.

2. Composition

   DetailPesanan menggunakan konsep komposisi dengan menampung object MenuItem sebagai bagian dari dirinya. MenuItem menjadi elemen wajib dalam sebuah DetailPesanan dan keberadaannya sangat bergantung pada object ini. Hal ini menunjukkan hubungan "bagian dari", di mana DetailPesanan tidak bisa berdiri tanpa item menu.
3. Polymorphism

   Atribut item bertipe MenuItem yang merupakan class induk bagi Makanan dan Minuman. Dengan cara ini, DetailPesanan dapat menerima berbagai jenis object yang berbeda dalam satu struktur yang sama. Method toString juga memanfaatkan polymorphism karena getInfo akan menghasilkan output sesuai jenis object menu.
4. Error Handling

   Constructor menyediakan beberapa validasi, seperti memastikan item tidak null dan jumlah pesanan harus lebih dari nol. Jika kondisi tidak terpenuhi, program melempar IllegalArgumentException untuk mencegah object dibuat dalam keadaan tidak valid. Catatan juga dibersihkan dari kemungkinan nilai null sehingga menghindari potensi error saat ditampilkan.

d. Makanan.java

Class Makanan digunakan untuk merepresentasikan menu makanan dalam sistem. Class ini menyimpan informasi seperti nama makanan, harga, tingkat pedas, dan kategori makanan. Class ini memperluas kemampuan dari MenuItem dengan menambahkan atribut khusus untuk makanan sehingga sistem dapat membedakan karakteristik antara makanan dan minuman.

Penerapan Konsep OOP

1. Inheritance

   Class ini mewarisi sifat dari MenuItem sehingga otomatis memiliki atribut dan method seperti nama dan harga. Dengan cara ini, kode menjadi lebih ringkas karena tidak perlu menulis ulang bagian yang sudah ada di class induk. Inheritance juga memperjelas bahwa makanan adalah salah satu jenis menu dalam sistem.
2. Encapsulation

   Atribut tingkatPedas dan kategori dibuat private sehingga hanya bisa diakses melalui getter. Hal ini menjaga data tetap aman dan mencegah perubahan langsung yang bisa merusak logika program. Konstruktor juga mengatur nilai awal dengan validasi sehingga setiap object Makanan selalu berada dalam kondisi valid.

3. Polymorphism

   Sebagai turunan MenuItem, object Makanan dapat diperlakukan sebagai MenuItem ketika digunakan di class lain seperti DetailPesanan atau Pesanan. Method toString juga merupakan bentuk polymorphism karena mengembalikan format yang berbeda dari class induknya.
4. Error Handling

   Constructor memeriksa apakah tingkat pedas berada dalam rentang yang benar dan memastikan kategori tidak kosong. Jika data tidak valid, program akan menolak object dengan melempar IllegalArgumentException. Validasi ini mencegah terjadinya data menu yang tidak logis seperti makanan tanpa kategori atau tingkat pedas negatif.

e. Meja.java

Class Meja digunakan untuk merepresentasikan meja yang ada di restoran. Setiap meja memiliki nomor dan status yang menunjukkan apakah meja tersebut sedang kosong atau terisi. Class ini membantu sistem melacak ketersediaan meja serta memastikan setiap pesanan terhubung ke meja yang valid.

Penerapan Konsep OOP

1. Encapsulation

   Atribut nomor dan status dibuat private sehingga hanya bisa diakses melalui getter dan setter. Mekanisme ini memastikan data meja tidak bisa diubah sembarangan dari luar class. Setter untuk status juga dilengkapi validasi agar tidak terjadi nilai status yang tidak logis.
2. Abstraction melalui Enum

   Status meja menggunakan enum StatusMeja yang berisi dua nilai, yaitu KOSONG dan TERISI. Enum mempermudah pengelolaan status karena nilainya sudah terbatas dan tidak bisa diubah menjadi nilai sembarangan. Ini membantu menjaga konsistensi data dalam sistem.
3. Error Handling

   Constructor memeriksa apakah nomor meja valid, yaitu harus lebih dari nol. Setter status juga memastikan nilai tidak null. Dengan cara ini, object Meja selalu dibuat dalam keadaan valid dan mencegah error seperti meja tanpa nomor atau status tidak dikenal.
4. Polymorphism (tidak eksplisit tetapi digunakan)

   Saat digunakan di class Pesanan, object Meja dapat diperlakukan sebagai tipe reference umum tanpa perlu mengetahui

detail internalnya. Hal ini merupakan bentuk penggunaan polymorphism dalam desain object.

f. MenuItem.java

Class MenuItem berfungsi sebagai blueprint dasar untuk semua jenis menu yang dijual di restoran. Class ini bersifat abstract sehingga tidak dapat dibuat objeknya secara langsung. MenuItem hanya menyediakan atribut umum seperti nama dan harga yang nantinya akan diwariskan ke class turunan seperti Makanan dan Minuman. Dengan cara ini, struktur menu menjadi lebih rapi dan konsisten.

Penerapan Konsep OOP

1. Inheritance

   MenuItem adalah class induk yang akan diwarisi oleh class lain. Makanan dan Minuman tidak perlu mendefinisikan ulang atribut nama atau harga, karena sudah tersedia di class ini. Hal ini mengurangi duplikasi kode dan membuat struktur lebih terorganisir.
2. Abstraction

   Karena class ini abstract, ia tidak mewakili objek menu tertentu. MenuItem hanya berfungsi sebagai kerangka umum. Abstraction membantu menyembunyikan detail yang tidak diperlukan dan membatasi pembuatan objek menu hanya melalui subclass yang lebih spesifik.
3. Encapsulation

   Atribut nama dan harga dibuat private sehingga aman dari manipulasi langsung. Akses dilakukan melalui getter yang terkontrol. Constructor juga memvalidasi nilai nama dan harga sebelum disimpan, menjaga agar setiap objek selalu berada dalam keadaan valid.
4. Polymorphism

   Class yang mewarisi MenuItem dapat diperlakukan sebagai tipe MenuItem, misalnya saat menyimpan menu dalam satu list yang berisi berbagai jenis makanan dan minuman. Ini memungkinkan sistem menangani objek berbeda melalui satu tipe referensi.

g. Minuman.java

Class Minuman adalah turunan dari MenuItem yang merepresentasikan menu minuman dengan atribut tambahan berupa ukuran dan suhu. Class ini memastikan bahwa setiap objek minuman memiliki informasi valid tentang jenis penyajiannya, seperti apakah minuman disajikan panas atau dingin, serta ukurannya. Class ini juga melakukan validasi agar tidak ada data kosong atau format suhu yang salah saat objek dibuat.

Penerapan Konsep OOP

1. Inheritance

   Class Minuman mewarisi atribut dan method dari MenuItem sehingga tidak perlu mendefinisikan ulang nama dan harga. Konsep ini membuat kode lebih ringkas dan menjaga agar struktur menu minuman mengikuti standar yang sama dengan menu lainnya.
2. Polymorphism

   Objek Minuman dapat diperlakukan sebagai objek MenuItem. Hal ini memudahkan sistem ketika menampilkan daftar menu atau menghitung total harga pesanan, karena makanan dan minuman dapat dikelola melalui satu tipe dasar yang sama.
3. Encapsulation

   Atribut ukuran dan suhu dibuat private sehingga hanya bisa diakses melalui getter. Constructor juga mengatur validasi agar ukuran tidak kosong dan suhu hanya boleh bernilai panas atau dingin. Dengan begitu, setiap objek Minuman selalu berada dalam kondisi data yang benar.
4. Abstraction

   Meskipun tidak langsung menggunakan keyword abstract, class Minuman berperan sebagai implementasi konkret dari MenuItem. MenuItem menyembunyikan detail umum menu, sementara Minuman mengisi detail spesifik yang hanya berlaku untuk minuman.

h. Pegawai.java

Class Pegawai berfungsi untuk merepresentasikan akun pegawai dalam sistem, seperti pelayan, koki, atau kasir. Class ini mewarisi struktur dasar dari Akun dan menambahkan atribut khusus berupa peran. Validasi ditambahkan agar setiap pegawai memiliki role yang benar dan konsisten, sehingga sistem dapat membedakan tanggung jawab setiap pegawai dengan jelas.

Penerapan Konsep OOP

1. Inheritance

   Pegawai mewarisi id, nama, dan password dari class Akun. Dengan konsep ini, seluruh data dasar pengguna tidak perlu ditulis ulang, dan semua jenis akun dalam sistem mengikuti struktur yang sama. Hal ini membuat kode lebih efisien dan mudah di-maintenance.
2. Polymorphism

   Karena Pegawai adalah turunan dari Akun, objek Pegawai dapat digunakan di tempat yang membutuhkan objek Akun. Sistem

dapat memperlakukan pegawai sebagai akun umum, tetapi tetap mempertahankan perilaku khusus yang dimiliki Pegawai.

3. Encapsulation

   Atribut peran dibuat private agar tidak bisa sembarangan diubah dari luar class. Aksesnya dikontrol melalui setter yang diberi validasi untuk memastikan nilai peran hanya boleh berupa pelayan, koki, atau kasir. Dengan cara ini, sistem tidak akan menerima data pegawai dengan peran yang tidak valid.

4. Abstraction

   Class Akun menyediakan struktur abstrak berupa identitas pengguna, sementara Pegawai melengkapi detail spesifik berupa peran dalam operasional restoran. Pengguna sistem tidak perlu tahu bagaimana validasi bekerja di dalamnya, tetapi hanya menggunakan objek Pegawai sesuai fungsinya.

i. Pesanan.java

Class Pesanan digunakan untuk merepresentasikan satu transaksi pemesanan yang dibuat pelanggan di restoran. Setiap pesanan memiliki ID unik, status, meja tempat pelanggan duduk, serta daftar item yang dipesan. Class ini juga menyediakan logika untuk menambah item ke pesanan dan menghitung total harga dari seluruh item yang dipesan. Validasi ditambahkan agar setiap pesanan hanya berisi data yang benar, misalnya ID positif, meja tidak null, status valid, dan detail pesanan tidak kosong.

Penerapan Konsep OOP

1. Encapsulation

   Atribut idPesanan, status, daftarItem, dan meja dibuat private dan hanya bisa diakses melalui getter, setter, atau method tertentu. Dengan cara ini, data dalam objek tidak bisa diubah sembarangan, sehingga integritas informasi pesanan tetap terjaga.

2. Abstraction

   Detail teknis seperti cara menghitung total harga, cara memvalidasi status, dan cara memeriksa validitas item disembunyikan di dalam method hitungTotal, setStatus, dan addDetail. Pengguna class cukup memanggil method tanpa melihat proses internalnya.

3. Composition

   Class Pesanan disusun dari objek DetailPesanan dan Meja. Pesanan tidak bisa berdiri sendiri tanpa kedua komponen tersebut. Ini menunjukkan hubungan kuat bahwa pesanan merupakan gabungan dari beberapa item pesanan dan satu meja pelanggan.

4. Polymorphism

   Daftar item di dalam Pesanan berisi objek DetailPesanan, yang

pada akhirnya mengacu ke objek MenuItem. MenuItem dapat berupa Makanan atau Minuman, tetapi Pesanan tidak perlu tahu jenis aslinya karena bisa mengakses harga dan jumlah melalui interface yang sama. Ini menunjukkan pemanfaatan polymorphism dalam alur pemesanan.

5. Error Handling

   Class ini melakukan pengecekan terhadap beberapa kondisi yang dapat menyebabkan error, seperti ID pesanan yang tidak valid, meja bernilai null, status yang tidak sesuai aturan, item pesanan yang null, atau total pesanan yang dihitung tanpa adanya item. Error handling ini menjaga agar object Pesanan tidak berada dalam keadaan yang salah atau tidak konsisten.

j. Struk.java

Class Struk digunakan untuk menampilkan hasil akhir transaksi dalam bentuk struk pembayaran. Class ini menerima sebuah objek Transaksi dan mencetak seluruh informasi penting seperti waktu transaksi, metode pembayaran, detail setiap item pesanan, subtotal, serta total keseluruhan. Class ini juga memastikan bahwa proses pencetakan hanya dapat dilakukan jika transaksi dan daftar pesanan valid sehingga struk yang dihasilkan tidak mengandung data yang salah.

Penerapan Konsep OOP

1. Encapsulation

   Class Struk menyimpan atribut transaksi sebagai private sehingga hanya dapat diakses melalui constructor atau method cetak. Dengan cara ini, informasi transaksi tidak bisa diubah dari luar class dan tetap terjaga konsistensinya. Akses ke data pesanan pun dilakukan melalui method getter, bukan langsung memanipulasi atribut internal.

2. Abstraction

   Class ini menyembunyikan detail bagaimana struk ditampilkan. Pengguna class hanya perlu memanggil method cetak tanpa harus mengetahui proses pengecekan, format output, atau cara menghitung subtotal setiap item. Semua kompleksitas tersebut ditangani secara internal.

3. Composition

   Struk bergantung pada objek Transaksi, dan transaksi itu sendiri berisi objek Pesanan serta daftar DetailPesanan. Ini menunjukkan hubungan composition karena Struk tidak dapat berfungsi tanpa keberadaan Transaksi yang valid.

4. Error Handling

   Class Struk melakukan pengecekan terhadap berbagai kondisi

yang dapat menyebabkan error, seperti transaksi yang bernilai null, pesanan yang tidak tersedia, atau daftar item yang kosong. Pada method cetak, proses pencetakan berada dalam blok try-catch untuk menangani error secara aman dan menampilkan pesan yang sesuai jika terjadi kegagalan. Dengan demikian, program tidak berhenti tiba-tiba ketika terjadi kondisi tidak valid.

k. Transaksi,java

Class Transaksi digunakan untuk merepresentasikan proses pembayaran dari sebuah pesanan. Pada saat objek Transaksi dibuat, class ini langsung memvalidasi pesanan, memastikan metode pembayaran benar, mencatat waktu transaksi, dan menghitung total harga secara otomatis berdasarkan seluruh item di dalam pesanan. Dengan kata lain, class ini menjadi penghubung antara pesanan yang sudah selesai dan proses pencetakan struk.

Penerapan Konsep OOP

1. Encapsulation

   Seluruh atribut seperti pesanan, total, metode pembayaran, dan waktu disimpan sebagai private sehingga tidak bisa diubah secara sembarangan dari luar. Akses informasi dilakukan melalui method getter, sedangkan perhitungan total dan validasi internal dilakukan dalam method khusus.
2. Abstraction

   Class ini menyembunyikan detail cara menghitung total, cara validasi metode pembayaran, dan proses pengecekan pesanan. Pengguna hanya perlu membuat objek Transaksi dan mengambil datanya tanpa tahu bagaimana semua proses internal tersebut bekerja.
3. Composition

   Transaksi memiliki hubungan erat dengan Pesanan, karena sebuah transaksi tidak dapat eksis tanpa pesanan yang valid. Transaksi juga menggunakan detail dari item-item di dalam pesanan untuk menghitung total. Ini menunjukkan hubungan “bagian dari” yang kuat.
4. Error Handling

   Class Transaksi melakukan berbagai bentuk pengecekan untuk mencegah data yang tidak valid. Pesanan tidak boleh null, pesanan harus memiliki item, metode pembayaran tidak boleh kosong, dan hanya bisa berupa cash, card, atau qris. Jika total hasil perhitungan tidak masuk akal, transaksi tidak diizinkan dibuat. Semua ini dilakukan untuk menjaga integritas data transaksi dan mencegah crash saat runtime.

l. User.java

Class yang digunakan untuk merepresentasikan akun login sederhana dalam sistem. Class ini menyimpan dua atribut utama yaitu username dan password yang harus diisi oleh pengguna yang ingin mengakses aplikasi. Class ini juga menyediakan validasi untuk mencegah data kosong atau tidak sesuai, sehingga sistem login menjadi lebih aman dan tidak menerima input sembarangan. Selain itu, class ini memiliki method login untuk mengecek apakah username dan password yang diberikan sesuai dengan data yang tersimpan, sehingga proses autentikasi dapat berjalan dengan baik. Class ini juga memiliki toString yang menampilkan username tanpa membocorkan password demi menjaga keamanan pengguna.

Penerapan Konsep OOP

1. Encapsulation

   Atribut username dan password dibuat private sehingga tidak bisa diakses langsung dari luar. Akses dan perubahan data hanya bisa dilakukan melalui setter dan getter yang sudah diberikan validasi. Encapsulation membantu menjaga data tetap aman dan mencegah perubahan sembarangan.
2. Abstraction

   Class ini menyembunyikan detail proses validasi dan pengecekan login dari pengguna lain. Pengguna cukup memanggil method login tanpa harus tahu bagaimana proses pengecekan dilakukan di dalam class. Abstraction memudahkan class digunakan tanpa perlu mengetahui detail internalnya.
3. Error handling

   Class ini menggunakan pemeriksaan input seperti pengecekan null, pengecekan input kosong, dan pengecekan panjang password minimal. Error handling dilakukan dengan memberikan exception yang jelas apabila terjadi kondisi tidak valid sehingga bug atau data rusak dapat dihindari sejak awal.

2.3 Service

Service berisi seluruh class yang mengatur logika inti sistem, yaitu bagaimana data dalam model digunakan dan bagaimana proses dalam aplikasi dijalankan. Jika folder model berfungsi untuk menyimpan struktur data dan representasi objek, maka folder service bertanggung jawab menjalankan proses bisnis seperti pembayaran, pengolahan pesanan, verifikasi, serta alur kerja aplikasi. Dengan kata lain, folder ini menjadi penghubung antara data dan alur operasional sistem.

Class pada folder service biasanya berisi operasi logis, baik melalui interface maupun class konkret. Interface digunakan untuk mendefinisikan kerangka perilaku yang bisa memiliki banyak bentuk implementasi, seperti berbagai jenis pembayaran. Class lainnya menangani pengelolaan sistem, misalnya memproses pesanan, menjalankan transaksi, atau mengatur mekanisme login. Karena fungsinya sebagai pusat logika, bagian service memegang peran penting dalam memastikan aplikasi berjalan sesuai aturan bisnis yang telah dirancang.

a. CardPayment.java

Class CardPayment adalah salah satu implementasi dari interface Pembayaran, yang digunakan untuk memproses pembayaran menggunakan kartu (debit/kredit). Class ini menangani input data kartu dari pengguna seperti nomor kartu dan nama pemilik, kemudian melakukan validasi untuk memastikan data yang dimasukkan benar sebelum pembayaran dianggap berhasil. Selain itu, class ini juga menghasilkan ID pembayaran unik menggunakan UUID.

Penerapan Konsep OOP

1. Encapsulation

   Seluruh atribut seperti idPembayaran, nomorKartu, dan namaPemilik disimpan sebagai private sehingga tidak dapat diakses langsung dari luar class. Akses hanya terjadi melalui method prosesPembayaran dan getter tertentu. Dengan cara ini, data sensitif seperti nomor kartu lebih terjaga dan tidak dapat dimodifikasi sembarangan.

2. Abstraction

   Class ini mengimplementasikan interface Pembayaran, yang hanya menyediakan kerangka umum berupa struktur method tanpa menjelaskan detail proses internalnya. CardPayment mengisi detail konkret seperti cara membaca input kartu, validasi nomor kartu, validasi nama pemilik, dan output hasil pembayaran. Abstraction membantu memisahkan logika umum dan logika spesifik.

3. Polymorphism

   Karena CardPayment mengimplementasikan interface Pembayaran, objek ini dapat diperlakukan sebagai tipe Pembayaran. Sistem dapat memanggil prosesPembayaran tanpa harus mengetahui bahwa implementasinya menggunakan kartu. Setiap metode pembayaran lain seperti cash atau qris dapat menggantikan CardPayment tanpa mengubah kode pemanggilnya.

4. Error Handling

   Class ini menggunakan pengecekan input yang ketat. Nomor kartu harus minimal 10 digit dan tidak boleh kosong, sedangkan nama pemilik juga wajib diisi. Jika input tidak valid, program melempar exception dan menampilkan pesan kesalahan yang jelas. Seluruh proses dibungkus dalam blok try-catch sehingga kesalahan tidak mematikan program dan pengguna tetap mendapat umpan balik yang benar.

b. CashPayment.java

   Class CashPayment digunakan untuk memproses pembayaran menggunakan uang tunai. Class ini meminta input jumlah uang yang diberikan pelanggan, melakukan pengecekan apakah uang tersebut cukup untuk membayar total pesanan, lalu menghitung kembalian jika ada. Selain itu, setiap transaksi tunai otomatis diberi id pembayaran unik menggunakan UUID. CashPayment menjadi implementasi konkret dari metode pembayaran tunai dalam sistem restoran.

Penerapan Konsep OOP

1. Encapsulation

   Atribut idPembayaran dan jumlahDibayar disimpan sebagai private sehingga tidak bisa diakses atau diubah secara langsung dari luar class. Informasi ini hanya dapat digunakan melalui method prosesPembayaran dan getter idPembayaran. Dengan cara ini, data pembayaran lebih aman dan tidak dapat dimanipulasi sembarangan.

2. Abstraction

   CashPayment mengimplementasikan interface Pembayaran yang hanya menyediakan struktur method umum. Detail cara meminta input uang, validasi jumlah, perhitungan kembalian, dan keluaran hasil pembayaran disembunyikan di dalam implementasi khusus class ini. Pengguna cukup memanggil prosesPembayaran tanpa memahami proses internalnya.

3. Polymorphism

   Karena CashPayment merupakan implementasi dari interface Pembayaran, objek ini dapat diperlakukan sebagai objek Pembayaran. Hal ini memungkinkan sistem untuk memanggil metode pembayaran secara generik tanpa perlu tahu apakah pengguna memilih cash, kartu, atau qris. Pemanggilan tetap sama, tetapi perilaku di dalamnya berbeda sesuai jenis pembayaran.

4. Error Handling

   Class ini memiliki pengecekan ketat terhadap input jumlah uang tunai. Jumlah uang tidak boleh nol atau negatif, dan tidak boleh kurang dari total pembayaran. Jika terjadi kondisi tidak valid, sistem melempar exception dan menampilkan pesan kesalahan yang jelas. Seluruh proses pembayaran dibungkus dalam blok try-catch untuk mencegah program berhenti ketika input salah.

c. Pembayaran.java

   Interface Pembayaran digunakan sebagai kontrak umum untuk seluruh jenis metode pembayaran dalam sistem. Interface ini menentukan dua hal penting yang wajib dimiliki setiap metode pembayaran, yaitu proses pembayaran itu sendiri dan ID pembayaran. Dengan adanya interface ini, sistem dapat memastikan bahwa semua jenis pembayaran, baik tunai, kartu, maupun QRIS, memiliki struktur dan cara pemanggilan yang konsisten.

Penerapan Konsep OOP

1. Abstraction

   Interface Pembayaran hanya berisi deklarasi method tanpa implementasi. Hal ini memberikan gambaran abstrak tentang apa yang harus dilakukan oleh setiap jenis pembayaran, tetapi tidak menentukan bagaimana caranya. Setiap class seperti CashPayment, CardPayment, dan QRISPayment bebas menentukan logic mereka sendiri saat mengimplementasikan interface ini.

2. Polymorphism

   Karena semua metode pembayaran mengikuti interface yang sama, sistem dapat memperlakukan seluruh jenis pembayaran sebagai objek Pembayaran. Hal ini memungkinkan RestaurantSystem memanggil prosesPembayaran secara seragam tanpa peduli apakah jenis pembayaran yang digunakan adalah cash, kartu, atau qris. Perilaku berbeda-beda, tetapi pemanggilannya tetap melalui tipe yang sama.

3. Encapsulation (tidak langsung)

   Walaupun interface tidak memiliki atribut, penggunaan interface membantu memaksa setiap class pembayaran menyembunyikan detail internal mereka. Class implementasi akan mengatur data secara private dan hanya memperlihatkan method yang telah ditentukan oleh interface, sehingga menjaga keteraturan dan keamanan struktur program.

d. QRISPayment.java

Class QRISPayment digunakan untuk merepresentasikan metode pembayaran menggunakan QRIS dalam sistem. Class ini mensimulasikan proses pembayaran digital, seperti menampilkan instruksi untuk melakukan scan QR dan menjalankan proses verifikasi otomatis. Hasil pembayaran dibuat acak untuk menggambarkan kemungkinan transaksi berhasil atau gagal, seperti kondisi yang sering terjadi pada pembayaran digital. Class ini juga menghasilkan ID pembayaran unik agar setiap transaksi QRIS dapat tercatat dengan jelas.

Penerapan Konsep OOP

1. Encapsulation

   Atribut idPembayaran disimpan sebagai private sehingga tidak dapat diakses atau diubah secara langsung dari luar class. Akses diberikan melalui method getter, sementara seluruh proses internal seperti hasil scan dan verifikasi tetap tersembunyi di dalam method prosesPembayaran.

2. Abstraction

   Class ini menyembunyikan detail proses pembayaran QRIS seperti simulasi waktu proses, pengecekan status transaksi, dan penanganan error. Pengguna cukup memanggil prosesPembayaran tanpa harus memahami bagaimana verifikasi QRIS dilakukan di dalam class. Hal ini membuat kode lebih sederhana dan mudah digunakan.

3. Polymorphism

   QRISPayment mengimplementasikan interface Pembayaran, sehingga dapat diperlakukan sebagai objek Pembayaran bersama CashPayment dan CardPayment. Pemanggilan proses pembayaran dapat dilakukan melalui tipe Pembayaran, dan masing-masing class memberikan perilaku berbeda sesuai jenis pembayarannya.

4. Error Handling

   Class ini menangani kemungkinan kegagalan transaksi dengan menggunakan try-catch. Jika simulasi pembayaran menghasilkan status gagal, program akan melempar exception dan memberi pesan error yang jelas. Dengan cara ini, sistem tetap berjalan meskipun QRISPayment mengalami kegagalan pembayaran.

e. RestaurantSystem.java

Class RestaurantSystem berfungsi sebagai pusat manajemen utama dari seluruh proses yang terjadi dalam aplikasi restoran. Class ini mengatur data menu, pegawai, user, customer, pesanan, serta proses pembayaran. Seluruh operasi penting seperti login, registrasi, penambahan data, pencarian

data, menampilkan menu, mengelola pesanan, dan memproses pembayaran dilakukan melalui class ini. Dengan kata lain, RestaurantSystem bertindak sebagai otak aplikasi yang menghubungkan berbagai komponen model dan service agar sistem dapat berjalan secara terstruktur.

Penerapan Konsep OOP

1. Encapsulation

   Semua daftar data seperti menu, pegawai, user, pesanan, dan customer disimpan sebagai atribut private. Akses ke seluruh data dilakukan melalui method khusus seperti getter, setter, atau method manipulasi data seperti tambahMenu, tambahPegawai, dan tambahPesanan. Dengan enkapsulasi ini, data penting tidak dapat dimanipulasi langsung dari luar class sehingga menjaga konsistensi dan keamanan sistem.

2. Abstraction

   Class ini menyembunyikan detail operasional dari proses restoran. Contohnya proses pembayaran, pencarian customer, pembuatan ID pesanan, serta validasi login. Pengguna cukup memanggil method sederhana seperti prosesPembayaran atau login tanpa perlu mengetahui bagaimana proses internal dilakukan. Abstraksi ini membuat kode lebih mudah digunakan, dipahami, dan dirawat.

3. Composition

   RestaurantSystem tersusun dari berbagai objek seperti MenuItem, Pegawai, User, Customer, dan Pesanan. Semua objek tersebut menjadi bagian penting dari RestaurantSystem dan tidak dapat berdiri sendiri dalam konteks sistem restoran. Komposisi ini menggambarkan hubungan “bagian dari” di mana sistem mengontrol siklus hidup data yang dikelolanya.

4. Polymorphism

   Class ini memanfaatkan polymorphism melalui interface Pembayaran. Saat proses pembayaran, RestaurantSystem dapat menerima berbagai jenis metode pembayaran seperti CashPayment, CardPayment, atau QRISPayment melalui satu tipe referensi Pembayaran. Hal ini memungkinkan sistem memproses pembayaran berbeda dengan cara yang sama tanpa harus membuat kode tambahan.

5. Error Handling

   Class RestaurantSystem mengelola banyak validasi dan penanganan error untuk menjaga sistem tetap stabil. Misalnya mencegah penambahan menu null, menangani login gagal, memeriksa

pesanan kosong, memastikan metode pembayaran valid, dan memberikan pesan error jika transaksi mengalami kegagalan. Dengan error handling ini, alur program tetap aman dan tidak mudah crash.

2.4 AppLauncher.Java

Class AppLauncher berfungsi sebagai titik awal saat aplikasi dijalankan. Class ini membuat objek RestaurantSystem sebagai pusat pengelolaan data, lalu memanggil LoginGUI untuk menampilkan tampilan login kepada pengguna. Dengan kata lain, AppLauncher adalah pintu masuk utama yang menghubungkan logic backend dengan tampilan GUI. Class ini hanya menjalankan proses inisialisasi awal tanpa menangani logika bisnis lain.

Penerapan Konsep OOP

1. Object Creation

   Class ini membuat objek RestaurantSystem sebagai instance utama yang dipakai di seluruh aplikasi. Pembuatan objek dilakukan satu kali sehingga dapat digunakan oleh GUI dan class lain. Hal ini menunjukkan penggunaan konsep object-oriented untuk mengelola sistem secara terstruktur.

2. Separation of Concerns

   AppLauncher memisahkan tugas secara jelas. Ia hanya menangani proses startup aplikasi, sementara logika aplikasi berada di RestaurantSystem dan tampilan di LoginGUI. Pemisahan ini membuat struktur sistem lebih rapi dan mudah dikelola.

3. Dependency Injection sederhana

   LoginGUI tidak membuat sistemnya sendiri, melainkan menerima objek RestaurantSystem yang dikirim dari AppLauncher. Cara ini membuat GUI tidak bergantung pada pembuatan instance baru, sehingga lebih fleksibel dan lebih mudah diuji.

2.5 Main.Java

Class Main adalah pusat alur program berbasis console dalam aplikasi ini. Class ini mengatur seluruh interaksi pengguna melalui terminal, mulai dari proses login, pembuatan akun, pembuatan pesanan, pengolahan pesanan oleh pegawai, hingga pembayaran. Main juga bertanggung jawab melakukan pemanggilan fungsi-fungsi utama yang dimiliki RestaurantSystem, sehingga class ini menjadi penghubung utama antara user dan seluruh fitur dalam sistem restoran. Selain itu, class Main menangani input pengguna, menampilkan menu navigasi, menyimpan data sederhana ke file, dan mengatur alur kerja berdasarkan peran pengguna seperti pelayan, koki, kasir, maupun customer.

Penerapan Konsep OOP

1. Abstraction

   Class Main menyembunyikan seluruh detail proses rumit di balik pemanggilan method dari RestaurantSystem dan class model lainnya. Pengguna hanya melihat tampilan menu dan memberikan input, sedangkan perhitungan total, validasi pesanan, pencarian data, serta logika bisnis dilakukan oleh class lain. Main hanya memanggil fungsi tanpa mengetahui seluruh detail implementasinya.

2. Encapsulation (melalui kerja sama dengan class lain)

   Meskipun Main tidak memiliki atribut khusus yang perlu dilindungi, ia bekerja dengan class lain yang menggunakan enkapsulasi. Main hanya bisa mengakses data melalui getter dan method tertentu, sehingga menjaga agar data internal objek seperti Pesanan, Customer, dan MenuItem tetap aman dan tidak berubah sembarangan. Main juga memanggil fitur-fitur sistem tanpa mengetahui struktur data di dalamnya.

3. Separation of Concerns

   Class Main hanya menangani alur aplikasi berbasis teks dan interaksi input-output. Logika bisnis seperti pembayaran, pembuatan pesanan, autentikasi, dan penyimpanan data ditangani oleh class lain seperti RestaurantSystem, Pembayaran, dan model. Pemisahan tugas ini membuat program lebih mudah dikelola dan tidak menumpuk seluruh logika di satu tempat.

4. Object Interaction

   Main adalah class yang paling banyak berinteraksi dengan objek lain. Ia membuat objek Pesanan, DetailPesanan, Customer, Pegawai, Meja, dan memanggil method pada objek-objek tersebut. Interaksi intensif ini menunjukkan cara class saling bekerja sama dalam OOP, di mana setiap class memiliki tanggung jawabnya sendiri dan dipanggil sesuai kebutuhan.

5. Error Handling

   Class ini menangani banyak kondisi error melalui pengecekan sederhana seperti validasi input, pengecekan data null, mencegah pesanan kosong, dan memastikan pemilihan menu valid. Dengan cara ini, program dapat tetap berjalan meskipun input pengguna tidak sesuai. Selain itu, Main memanggil method dari class lain yang memiliki exception dan error handling internal untuk mencegah crash selama runtime.

C. Kesimpulan

Berdasarkan implementasi yang telah dilakukan pada project UAS ini, dapat disimpulkan bahwa konsep dasar pemrograman berorientasi objek telah diterapkan secara menyeluruh pada setiap bagian sistem. Struktur project yang terbagi menjadi model, service, GUI, dan class utama menunjukkan bahwa perancangan dilakukan secara modular sehingga setiap komponen memiliki tanggung jawab yang jelas. Konsep OOP seperti encapsulation, inheritance, polymorphism, abstraction, dan composition diterapkan pada hampir seluruh class untuk menjaga keamanan data, memperjelas relasi antar objek, serta mempermudah pengembangan sistem. Setiap class juga dilengkapi dengan mekanisme error handling agar aplikasi lebih aman, tidak mudah mengalami crash, dan mampu menangani input yang tidak valid. Dengan desain ini, sistem pemesanan restoran menjadi lebih terstruktur, mudah dipahami, dan dapat dikembangkan kembali di masa mendatang. Project ini juga menunjukkan bagaimana konsep OOP tidak hanya menjadi teori, tetapi dapat digunakan sebagai dasar dalam membangun aplikasi yang stabil, rapi, dan realistis sesuai kebutuhan.