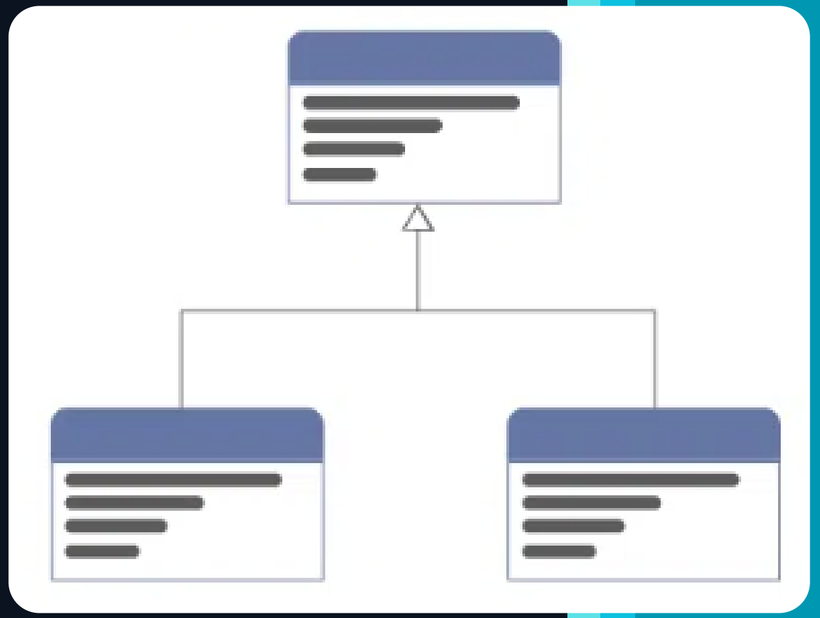
CLASS DIAGRAM

Kelompok 4



RINCIAN MATERI

1. Definisi Class Diagram
2. Tujuan atau Kegunaan Class Diagram
3. Perbandingan Class Diagram dengan Activity Diagram dan Use Case Diagram
4. Access Modifire Class Diagram
5. Komponen Class Diagram
6. Contoh Kasus Penggambaran Class Diagram

TUJUAN PEMBELAJARAN

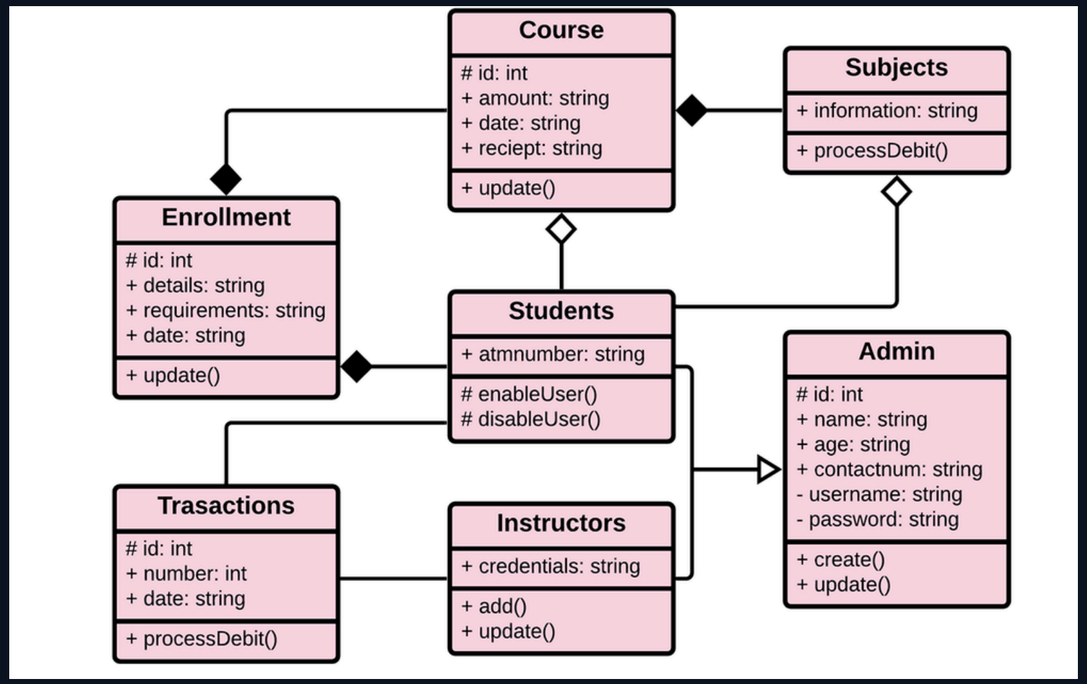
1. Audiens mampu memahami Class Diagram
2. Audiens mampu membedakan Class Diagram, Activity Diagram, dan Use Case Diagram
3. Audiens mampu menggunakan Class Diagram

PRE-TEST

APA ITU CLASS DIAGRAM?

APA ITU CLASS DIAGRAM?

Class Diagram adalah salah satu jenis diagram dalam Unified Modeling Language (UML) yang digunakan untuk merepresentasikan struktur statis dari sebuah sistem berorientasi objek. Diagram ini menggambarkan kelas-kelas dalam sistem, atribut dan operasi yang dimiliki oleh setiap kelas, serta hubungan antar kelas.



Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). The Unified Modeling Language Reference Manual (2nd ed.). Addison-Wesley Professional

APA ITU OBJEK & CLASS

Class adalah konsep dasar dalam berbasis OOP yang berfungsi sebagai **cetak biru** untuk menciptakan objek.

Objek adalah **hasil cetak dari class** atau hasil ‘konkrit’ dari class

Contoh Class :

```php
<?php
class laptop {
   // isi dari class laptop...
}
?>
```

Contoh Object :

```php
<?php
class laptop {
   //... isi dari class laptop
   }

$laptop_andi = new laptop();
$laptop_anto = new laptop();
?>
```

HUBUNGAN OBJEK & CLASS

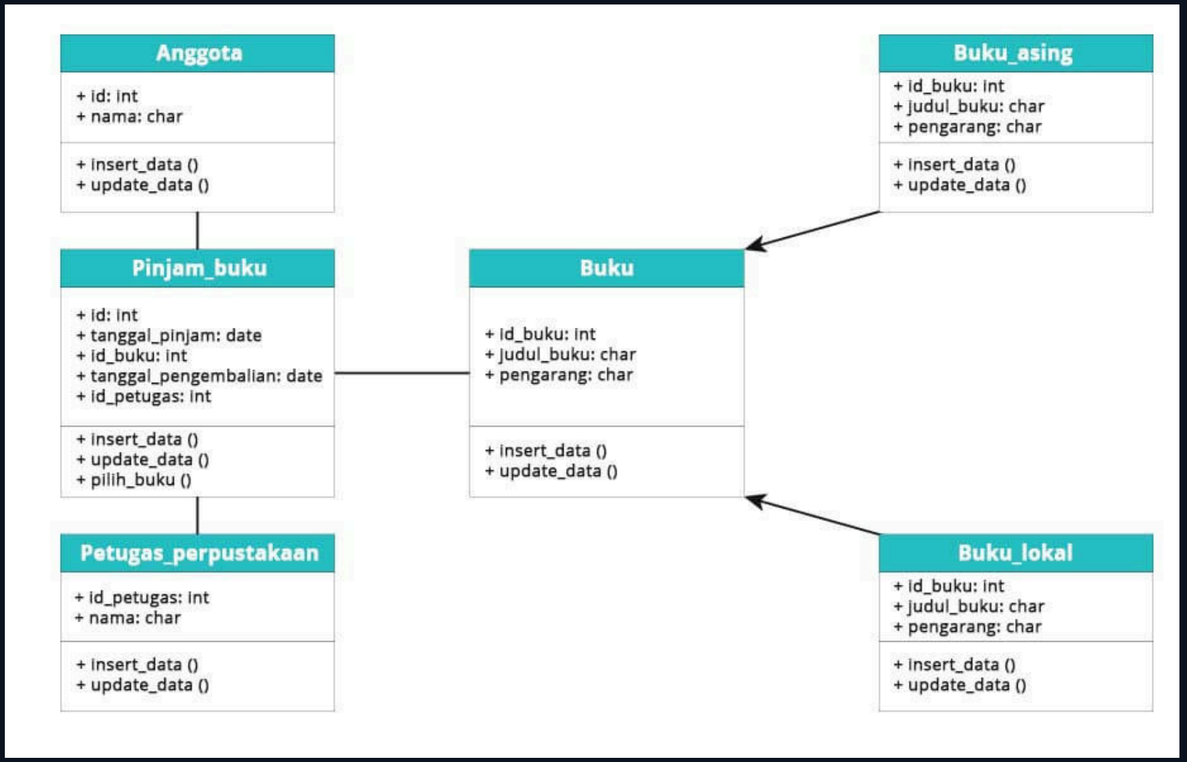
Hubungan antara class dan objek dapat di jelaskan dengan beberapa poin poin berikut:

- Satu class dapat memberikan dasar untuk menciptakan banyak objek yang berbeda-beda.
- Setiap objek yang dibuat akan memiliki karakteristik, perilaku, dan atribut yang unik.
- Perubahan yang dilakukan pada class akan berdampak pada semua objek yang sudah dibuat sebelumnya. Namun, objek yang dibuat setelah perubahan tidak akan terpengaruh.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

TUJUAN DAN FUNGSI CLASS DIAGRAM

Tujuan class diagram



Tujuan utama dari class diagram adalah untuk menyajikan gambaran umum dari struktur sistem perangkat lunak, memvisualisasikan hubungan antar kelas, dan membantu dalam pengembangan sistem yang lebih efisien dan efektif.

Diagram ini sangat penting dalam perancangan sistem berbasis objek, karena dapat membantu memahami konsep dasar seperti pewarisan, enkapsulasi, dan polimorfisme.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

Fungsi class diagram

1. Memahami Struktur Sistem
2. Mengembangkan Secara Kolaboratif
3. Merencanakan dan Merancang Sistem
4. Mendokumentasikan Sistem
5. Mengoptimalkan dan Memperbaiki Sistem

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

PERBANDINGAN CLASS DIAGRAM DENGAN ACTIVITY DIAGRAM DAN USE CASE DIAGRAM

CLASS DIAGRAM

- Fokus: Menggambarkan struktur statis dari sebuah sistem. Class Diagram menampilkan kelas-kelas (class), atribut, metode, serta hubungan antar kelas.
- Kegunaan: Sangat berguna dalam memodelkan sistem berorientasi objek, memungkinkan Anda untuk melihat bagaimana entitas yang berbeda berinteraksi satu sama lain di level data. Ini mencakup pewarisan (inheritance), asosiasi, agregasi, dan komposisi.
- Contoh Penggunaan: Saat mengembangkan perangkat lunak berbasis objek, seperti dalam desain aplikasi atau sistem informasi.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

ACTIVITY DIAGRAM

- Fokus: Menggambarkan perilaku dinamis dari sebuah sistem, khususnya alur aktivitas atau proses bisnis. Diagram ini menunjukkan alur langkah demi langkah (sekuensial atau paralel), termasuk kondisi logika dan keputusan.
- Kegunaan: Bermanfaat untuk menggambarkan logika proses atau algoritma, serta untuk memahami bagaimana sistem akan menjalankan tugas atau alur kerja tertentu. Ini juga sering digunakan untuk pemodelan proses bisnis.
- Contoh Penggunaan: Memvisualisasikan alur kerja dalam sistem pemrosesan pesanan atau algoritma dalam sebuah perangkat lunak.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

USE CASE DIAGRAM

- Fokus: Menggambarkan interaksi antara pengguna luar (aktor) dengan sistem, dengan menekankan pada fungsi-fungsi utama dari perspektif pengguna.
- Kegunaan: Diagram ini sangat cocok untuk menggambarkan kebutuhan (requirements) dan skenario penggunaan, memberi pandangan sederhana mengenai bagaimana aktor eksternal terlibat dalam sistem.
- Contoh Penggunaan: Dalam tahap awal pengembangan perangkat lunak untuk memetakan fungsionalitas yang dibutuhkan oleh berbagai pemangku kepentingan atau pengguna sistem.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

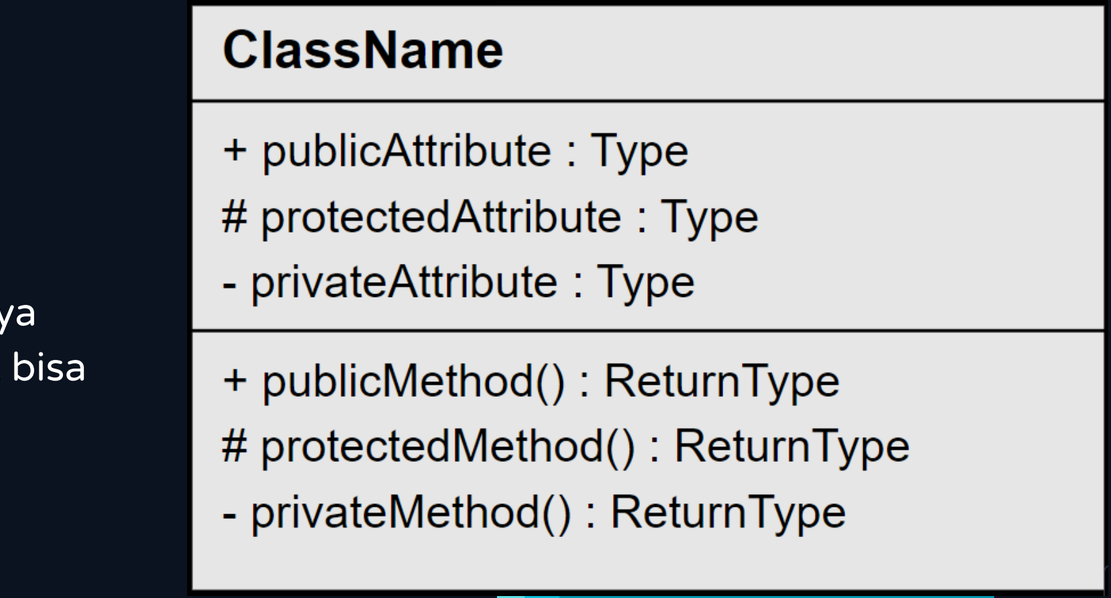
APAKAH DI CLASS DIAGRAM ADA ACCESS MODIFIRE?

ACCESS MODIFIRE CLASS DIAGRAM

1. ## Public
Atribut atau metode dengan access modifier public dapat diakses oleh kelas lain tanpa batasan.

2. ## Private
Atribut atau metode dengan access modifier private hanya dapat diakses dari dalam kelas itu sendiri. Kelas lain tidak bisa mengakses atribut atau metode ini secara langsung.

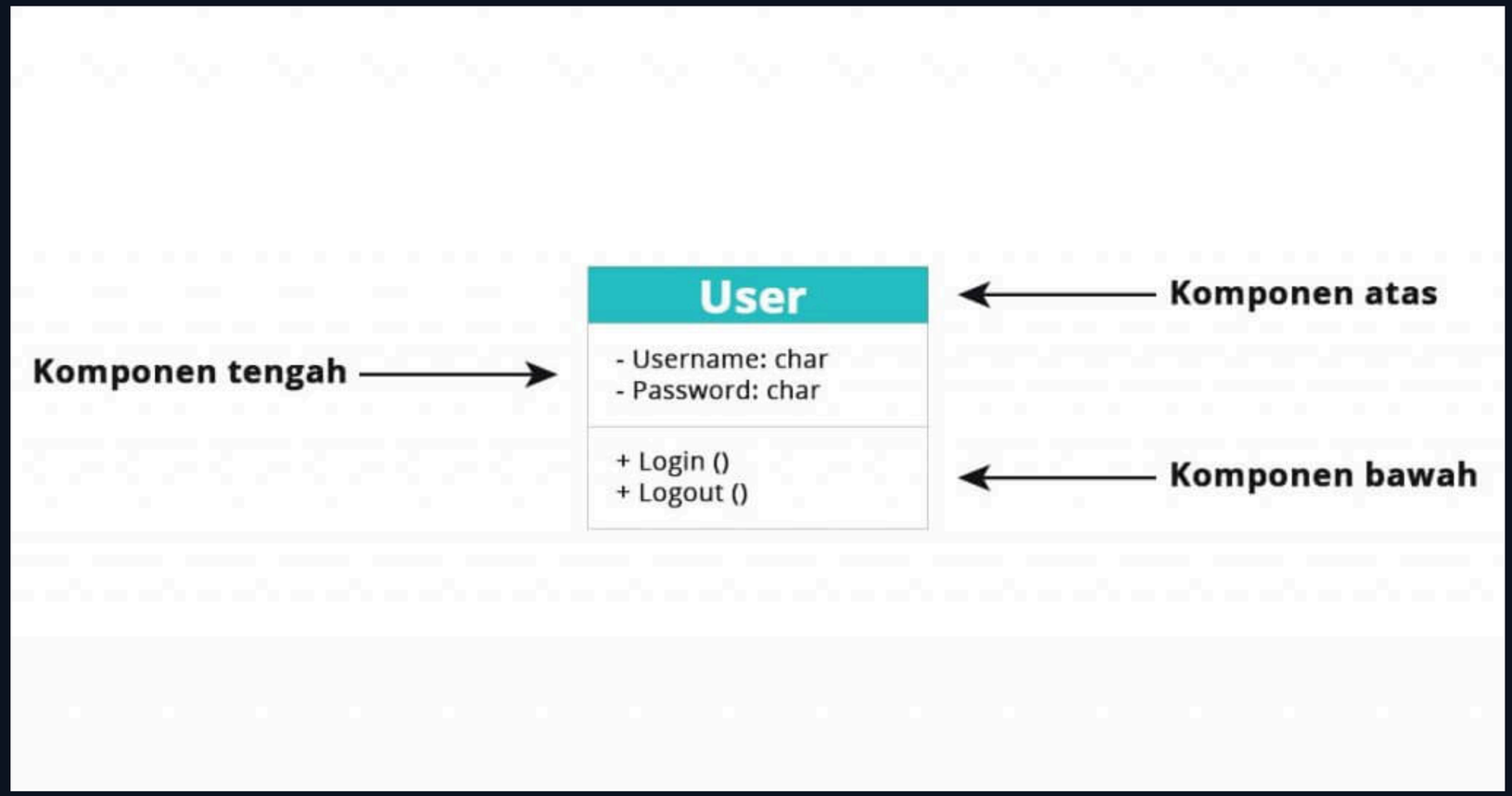
3. ## Protected
Atribut atau metode yang bersifat protected dapat diakses oleh kelas itu sendiri, subclass (kelas turunan), atau kelas dalam yang sama.



Rumbaugh, J., Jacobson, I., & Booch, G. (2004). The Unified Modeling Language Reference Manual (2nd ed.). Addison-Wesley Professional

KOMPONEN CLASS DIAGRAM

KOMPONEN CLASS DIAGRAM



KOMPONEN CLASS DIAGRAM

1

Komponen Atas

Komponen ini berisikan nama class. Setiap class pasti memiliki nama yang berbeda-beda, sebutan lain untuk nama ini adalah simple name (nama sederhana).

2

Komponen Tengah

Komponen ini berisikan atribut dari class, komponen ini digunakan untuk menjelaskan kualitas dari suatu kelas. Atribut ini dapat menjelaskan dapat ditulis lebih detail, dengan cara memasukan tipe nilai.

3

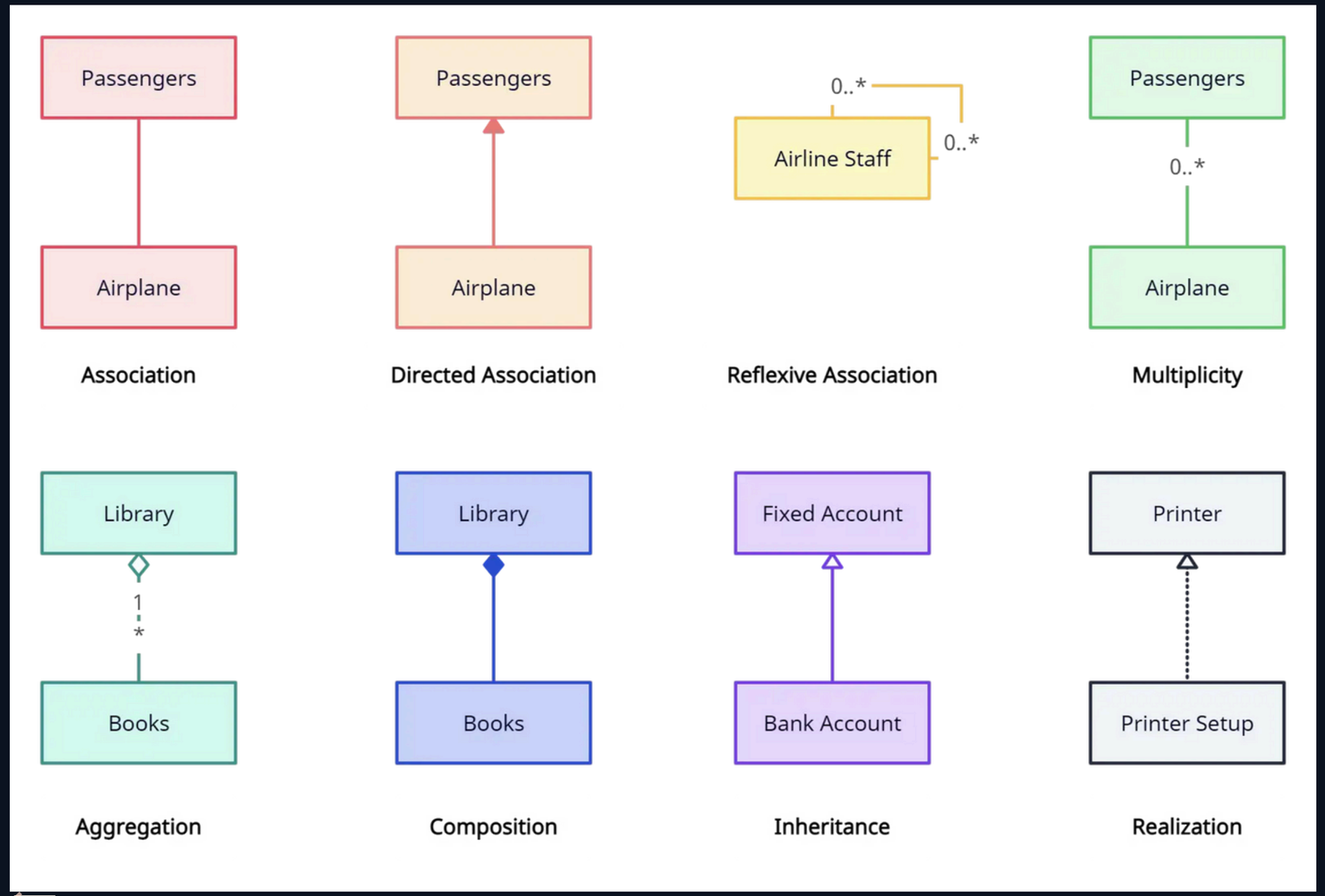
Komponen Bawah

Komponen ini menyertakan operasi yang ditampilkan dalam bentuk daftar. Operasi ini dapat menggambarkan bagaimana suatu class dapat berinteraksi dengan data.

Dicoding

HUBUNGAN ANTAR CLASS

HUBUNGAN ANTAR CLASS



visual paradigm

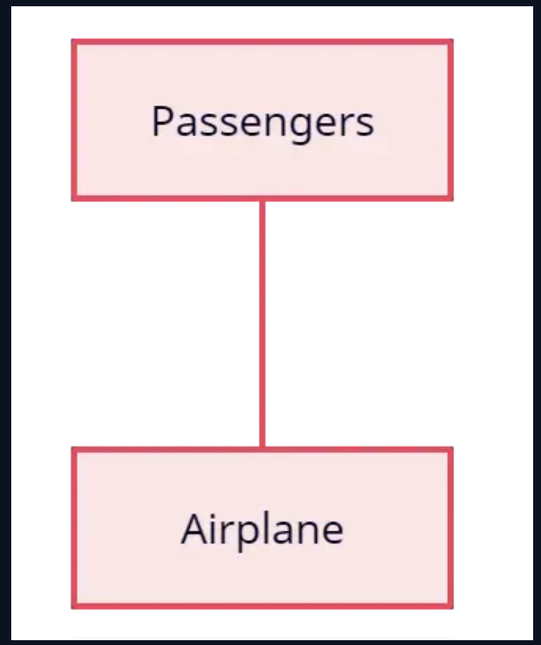
HUBUNGAN ANTAR CLASS

1. Association (Asosiasi)
2. Directed Association (Asosiasi Terarah)
3. Reflexive Association (Asosiasi Reflektif)
4. Multiplicity (Kardinalitas)
5. Aggregation (Agregasi)
6. Composition (Komposisi)
7. Inheritance/Generalization (Pewarisan/Generalisasi)
8. Realization (Realisasi)

HUBUNGAN ANTAR CLASS

1 Association (Asosiasi)

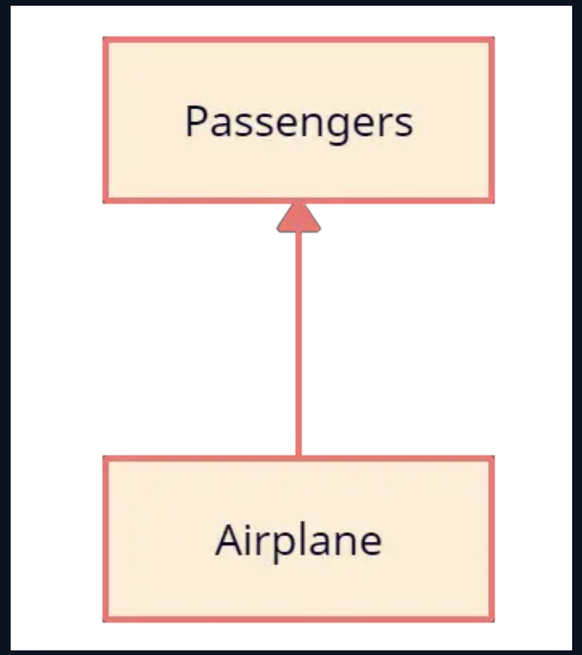
Asosiasi adalah Hubungan antara dua kelas yang menunjukkan bahwa satu objek dari kelas pertama berhubungan dengan satu atau lebih objek dari kelas kedua. Sebagai contoh, penumpang dan maskapai penerbangan mungkin dihubungkan seperti yang ditunjukkan di atas.



HUBUNGAN ANTAR CLASS

2 Directed Association (Asosiasi Terarah)

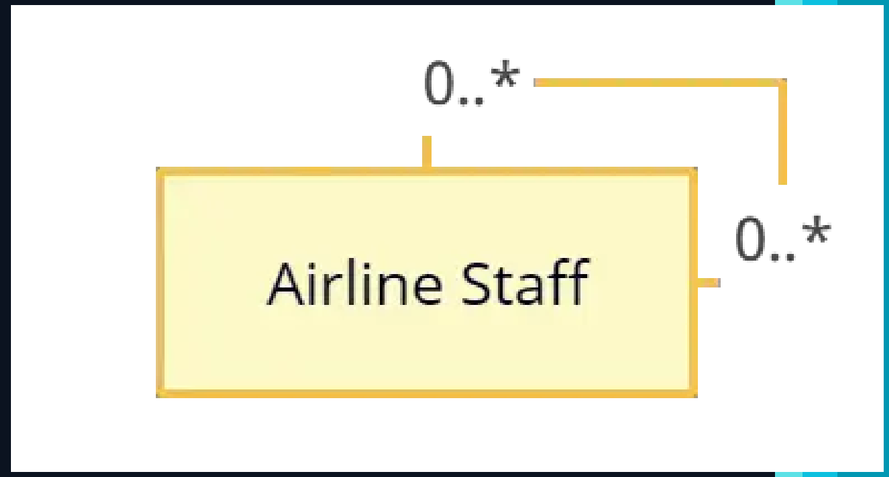
Asosiasi terarah Mirip dengan asosiasi, tetapi memiliki arah yang jelas. Panah menunjukkan arah hubungan dari satu kelas ke kelas lainnya. Sebagai contoh, kelas Penumpang mungkin memiliki hubungan terarah dengan kelas Penerbangan.



HUBUNGAN ANTAR CLASS

3 Reflexive Association (Asosiasi Reflektif)

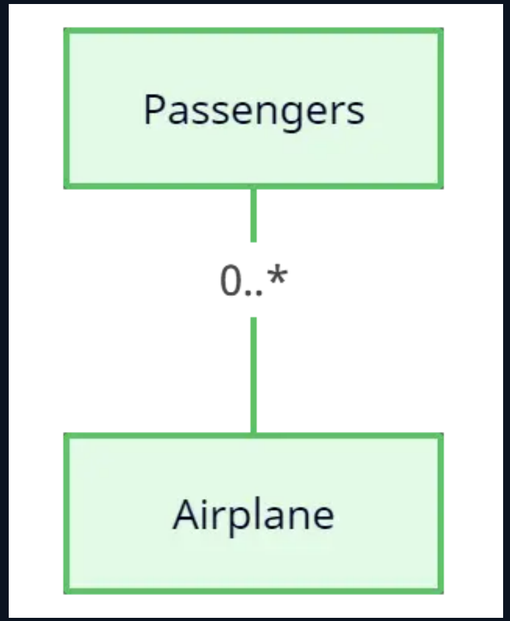
Ini terjadi ketika suatu kelas berhubungan dengan dirinya sendiri. Sebagai contoh, seorang staf di bandara bisa menjadi pilot, insinyur penerbangan, atau petugas tiket. Jika seorang insinyur penerbangan mengelola anggota staf pemeliharaan, bisa ada hubungan dikelola oleh antara dua instance dari kelas yang sama.



HUBUNGAN ANTAR CLASS

4 Multiplicity (Kardinalitas)

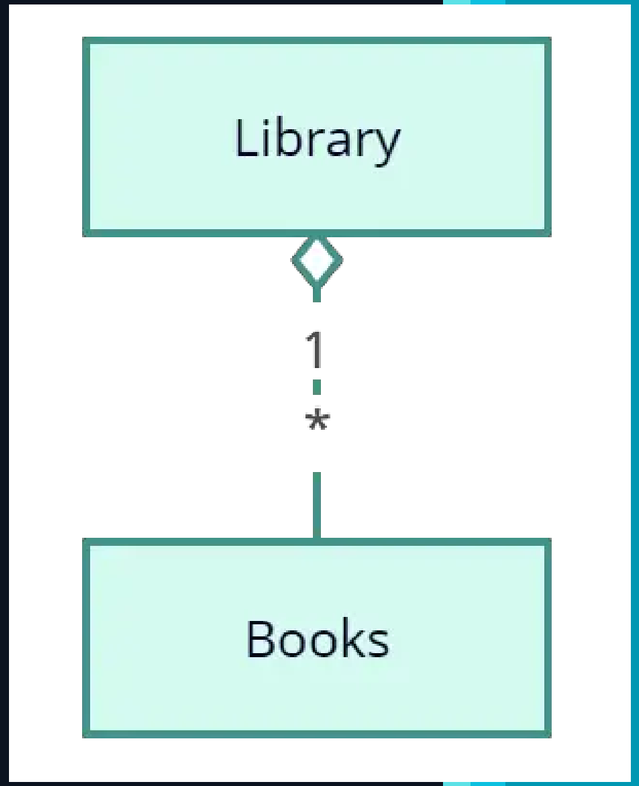
Kardinalitas Menunjukkan jumlah objek dari satu kelas yang terkait dengan objek dari kelas lain. Sebagai contoh, satu armada bisa terdiri dari banyak pesawat, dan satu pesawat bisa membawa nol hingga banyak penumpang. Notasi "0..*" menunjukkan "nol hingga banyak" dalam diagram.



HUBUNGAN ANTAR CLASS

5 Aggregation (Agregasi)

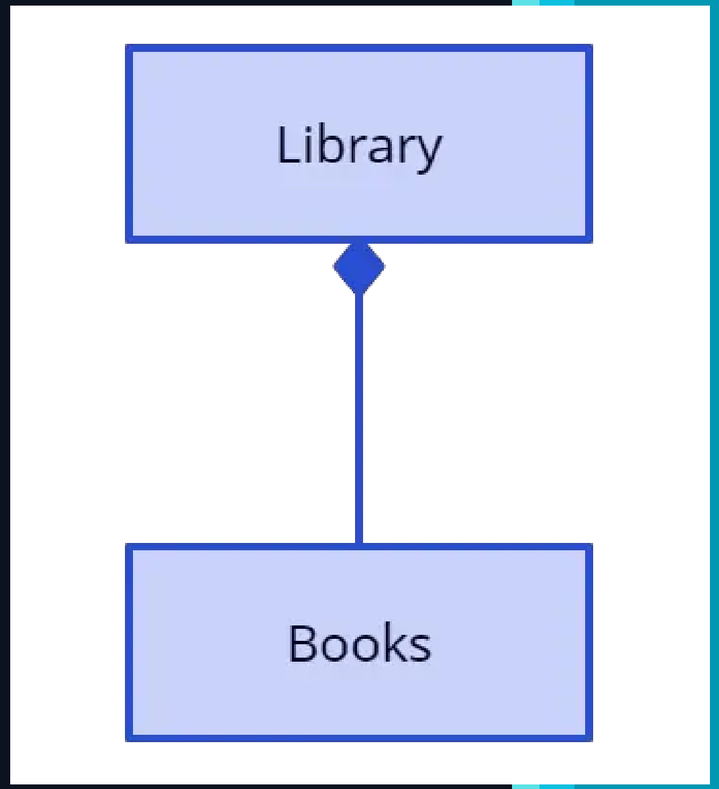
Agregasi adalah Hubungan "bagian-dari" yang menunjukkan bahwa satu kelas terdiri dari beberapa komponen yang masih dapat eksis sendiri. Sebagai contoh, kelas Perpustakaan terdiri dari satu atau lebih Buku. Dalam agregasi, kelas-kelas yang terkandung tidak bergantung secara kuat pada siklus hidup kelas kontainer. Buku akan tetap ada meskipun perpustakaan ditutup. Dalam diagram UML, hubungan agregasi ditunjukkan dengan garis dan bentuk belah ketupat kosong di dekat kelas induk.



HUBUNGAN ANTAR CLASS

6 Composition (Komposisi)

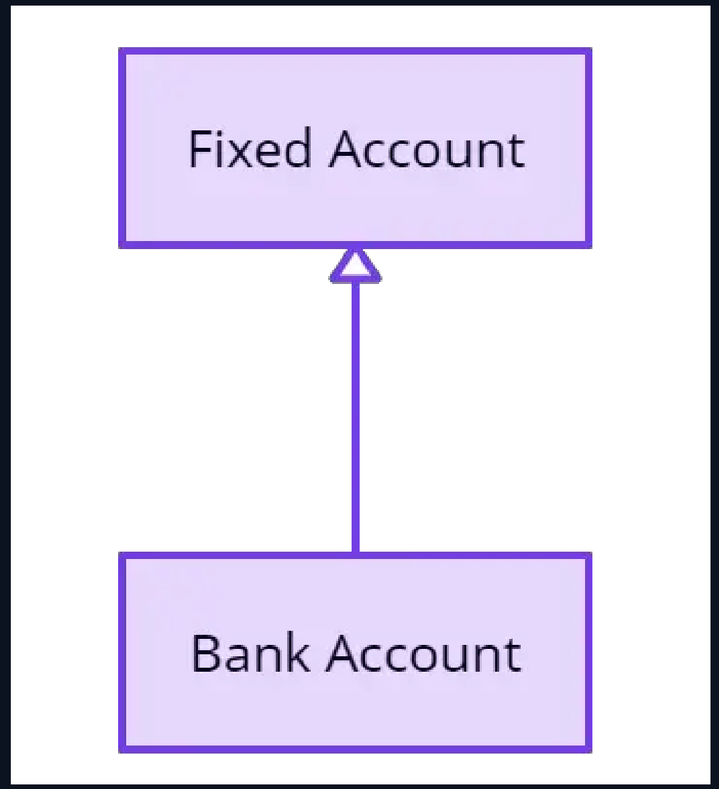
Hubungan komposisi Hubungan "bagian-dari" yang lebih kuat, di mana bagian tidak dapat eksis tanpa keseluruhannya. Sebagai contoh, kantong samping tas punggung akan hancur jika tasnya dihancurkan. Dalam diagram UML, hubungan komposisi ditunjukkan dengan garis dan bentuk belah ketupat penuh di dekat kelas induk.



HUBUNGAN ANTAR CLASS

7 Inheritance/Generalization (Pewarisan/Generalisasi)

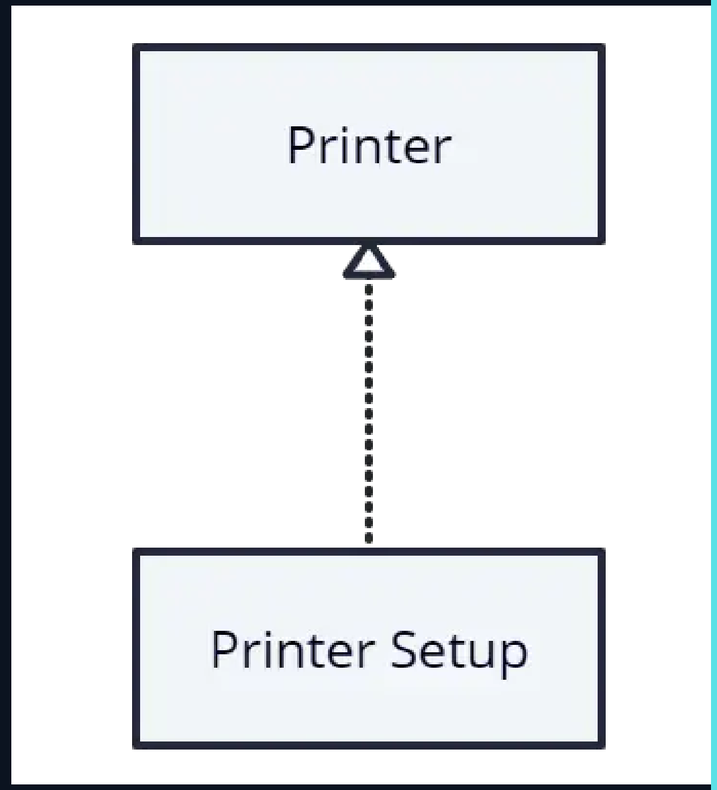
Pewarisan adalah Menunjukkan bahwa satu kelas adalah versi khusus dari kelas lain. Untuk menunjukkan pewarisan dalam diagram UML, garis solid ditarik dari kelas anak ke kelas induk dengan panah kosong.



HUBUNGAN ANTAR CLASS

8 Realization (Realisasi)

Realisasi Menunjukkan bahwa satu kelas menyediakan implementasi dari antarmuka atau fungsionalitas yang didefinisikan oleh kelas lain. Dalam diagram UML, hubungan ini ditunjukkan dengan garis putus-putus dan panah kosong dari kelas yang mendefinisikan fungsionalitas ke kelas yang mengimplementasikannya. Sebagai contoh, preferensi pencetakan yang diatur melalui antarmuka setup printer diimplementasikan oleh printer.



KEGUNAAN, KELEBIHAN DAN KEKURANGAN PADA RELASI CLASS DIAGRAM

1 Association (Asosiasi)

Kegunaan: Menggambarkan hubungan umum antara dua kelas yang terhubung secara logis. Misalnya, hubungan antara Penumpang dan Penerbangan.

- Kelebihan:
  - Mudah dipahami dan fleksibel untuk menggambarkan berbagai hubungan antar kelas.
  - Memungkinkan penambahan kardinalitas untuk menunjukkan jumlah instance yang terlibat.
- Kekurangan:
  - Tidak menunjukkan hierarki atau ketergantungan siklus hidup antara objek-objek.

2 Directed Association (Asosiasi Terarah)

Kegunaan: Menunjukkan arah hubungan dari satu kelas ke kelas lain, menunjukkan mana yang lebih dominan dalam hubungan tersebut.

- Kelebihan:
  - Membantu memperjelas aliran atau arah kontrol antar objek.
  - Memudahkan untuk memahami kelas mana yang menjadi sumber atau tujuan hubungan.
- Kekurangan:
  - Membatasi fleksibilitas jika suatu saat hubungan berubah menjadi dua arah.

3 Reflexive Association (Asosiasi Reflektif)

Kegunaan: Menunjukkan bahwa satu kelas bisa memiliki hubungan dengan dirinya sendiri. Misalnya, dalam manajemen organisasi, seorang karyawan bisa mengelola karyawan lain dalam struktur yang sama.

- Kelebihan:
  - Memungkinkan penggambaran hubungan kompleks dalam satu kelas.
  - Berguna untuk sistem hirarkis atau struktur organisasi.
- Kekurangan:
  - Bisa membingungkan untuk dibaca jika terlalu banyak hubungan reflektif dalam satu diagram.

4 Multiplicity (Kardinalitas)

Kegunaan: Menjelaskan jumlah instance yang terlibat dalam sebuah hubungan. Misalnya, satu Perusahaan bisa memiliki banyak Karyawan.

- Kelebihan:
  - Membantu memperjelas hubungan kuantitatif antar objek.
  - Memberikan kendali lebih dalam pemodelan sistem yang kompleks.
- Kekurangan:
  - Dapat memperumit diagram jika terlalu banyak detail kardinalitas yang ditambahkan.

5 Aggregation (Agregasi)

Kegunaan: Menunjukkan hubungan "whole-part", di mana satu kelas terdiri dari beberapa bagian tetapi bagian tersebut bisa berdiri sendiri. Misalnya, Perpustakaan memiliki Buku.

- Kelebihan:
  - Memudahkan representasi hubungan komponen yang independen.
  - Tidak bergantung pada siklus hidup kelas induk, jadi lebih fleksibel.
- Kekurangan:
  - Tidak menunjukkan ketergantungan yang kuat antara kelas induk dan komponen.

6 Composition (Komposisi)

Kegunaan: Mirip dengan agregasi, tetapi bagian-bagian tidak bisa hidup tanpa induknya. Misalnya, Rumah dan Kamar.

- Kelebihan:
  - Menggambarkan ketergantungan siklus hidup yang kuat, jelas menunjukkan bahwa bagian-bagian akan dihancurkan bersama kelas induknya.
  - Berguna dalam model yang sangat terstruktur.
- Kekurangan:
  - Terlalu ketat, karena kelas yang dikomposisikan akan hancur bersama induknya, membatasi fleksibilitas desain.

7 Inheritance/Generalization (Pewarisan/Generalisasi)

Kegunaan: Menunjukkan hubungan “is-a” di mana satu kelas adalah turunan dari kelas lainnya. Misalnya, Hewan dan Anjing.

- Kelebihan:
  - Memfasilitasi penggunaan kembali kode, karena anak kelas mewarisi sifat-sifat dari kelas induk.
  - Memungkinkan polimorfisme dan abstraksi dalam desain.
- Kekurangan:
  - Dapat menyebabkan desain yang terlalu kaku dan hirarkis jika digunakan berlebihan.
  - Pewarisan terlalu dalam (deep inheritance) bisa mempersulit pemeliharaan kode.

8 Realization (Realisasi)

Kegunaan: Menunjukkan hubungan antara interface (kontrak) dan implementasinya. Misalnya, kelas Printer mengimplementasikan interface Printable.

- Kelebihan:
  - Membantu dalam desain berbasis kontrak dan memungkinkan modularitas tinggi.
  - Mendorong penggunaan antarmuka, yang memungkinkan fleksibilitas dan mudah diubah.
- Kekurangan:
  - Mungkin lebih sulit dipahami oleh pemula karena pemisahan antara kontrak dan implementasi.

STUDI KASUS

STUDI KASUS

Studi Kasus: Sistem Manajemen Restoran

Sebuah restoran ingin mengembangkan sistem manajemen untuk mengelola data menu, pesanan pelanggan, dan staf. Sistem ini akan mencatat detail makanan dan minuman yang tersedia, informasi pesanan yang dilakukan oleh pelanggan, serta data staf yang bertugas.

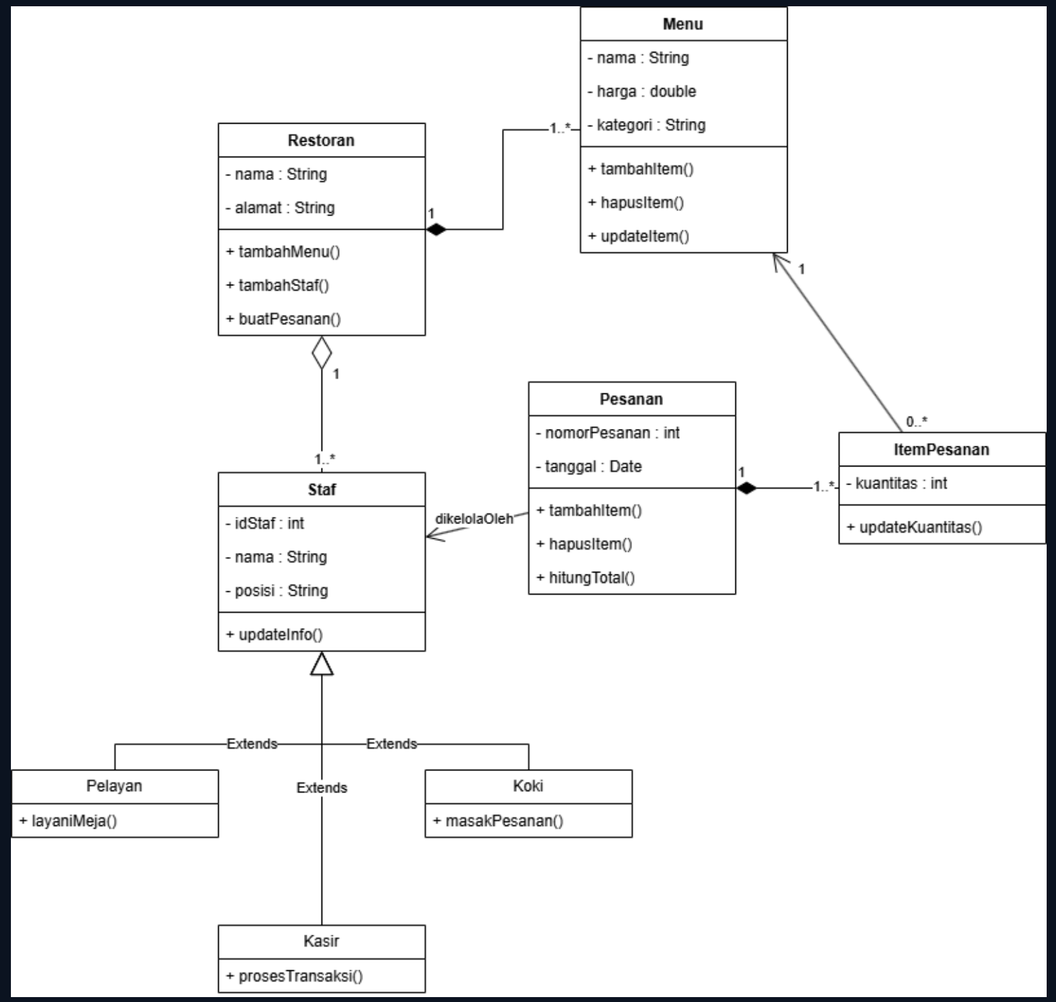
Kebutuhan Sistem

Menu: Sistem harus mencatat detail menu seperti nama item, harga, dan kategori (makanan atau minuman).

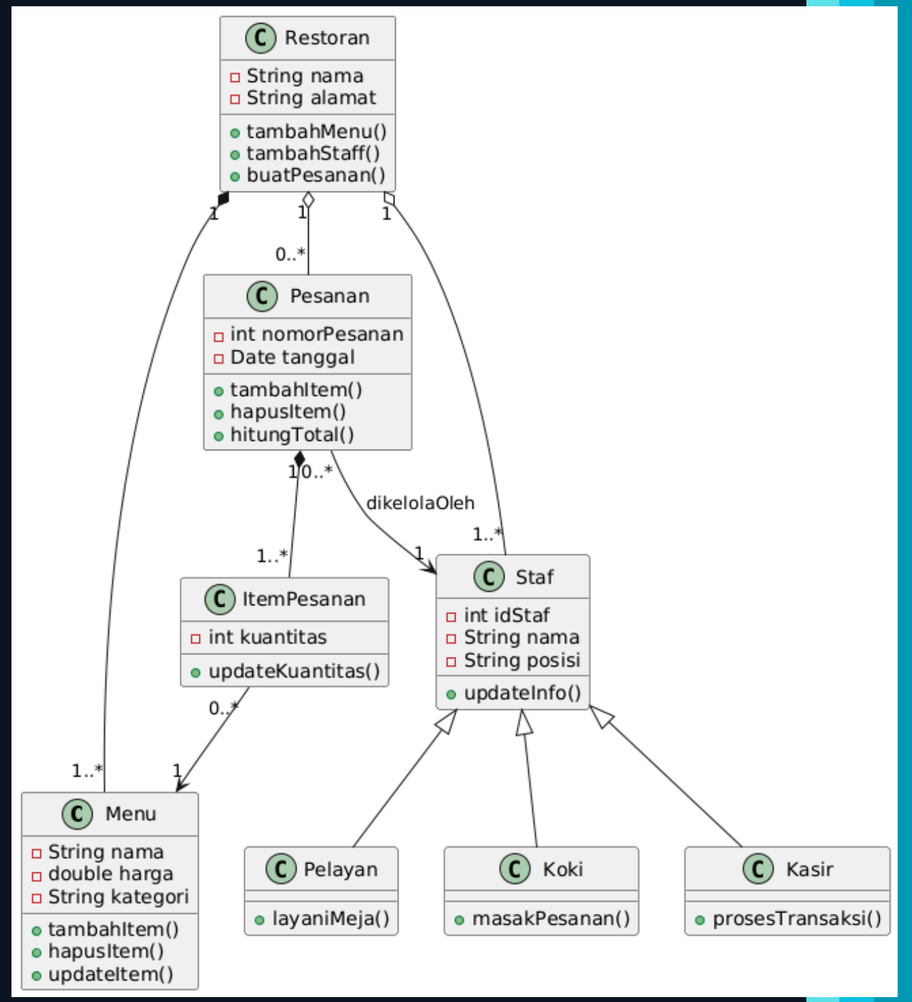
Pesanan: Sistem harus mencatat pesanan yang dilakukan pelanggan, termasuk nomor pesanan, tanggal, dan item yang dipesan beserta kuantitasnya.

Staf: Sistem harus menyimpan data staf yang bertugas, seperti ID staf, nama, dan posisi (pelayan, koki, atau kasir).

STUDI KASUS



Draw.io



Plant UML

POST-TEST

KESIMPULAN

Class Diagram dalam Unified Modeling Language (UML) adalah alat penting untuk menggambarkan struktur statis dari sistem berbasis objek, termasuk kelas, atribut, operasi, dan hubungan antar kelas. Diagram ini memungkinkan pemahaman yang lebih baik mengenai hubungan seperti asosiasi, agregasi, komposisi, dan pewarisan, yang masing-masing memiliki kegunaan, kelebihan, dan kekurangan tertentu dalam memodelkan sistem. Asosiasi menunjukkan hubungan umum antar kelas, sementara agregasi dan komposisi menyoroti hubungan "whole-part" dengan tingkat ketergantungan yang berbeda. Pewarisan memungkinkan penggunaan kembali kode dan polimorfisme, namun bisa menyebabkan desain yang kaku jika digunakan berlebihan. Dengan memahami hubungan-hubungan ini, class diagram membantu merancang sistem yang lebih terstruktur, modular, dan fleksibel, tetapi juga harus diwaspadai agar tidak memperumit diagram dengan terlalu banyak detail atau hierarki yang mendalam.