💡 Pengertian Dependency

Dependency adalah **relasi semantik** antara dua elemen model, yang menunjukkan bahwa **elemen yang bergantung (dependent)** memerlukan **elemen lain (independent)** untuk berfungsi, dimengerti, atau diimplementasikan dengan benar.

Artinya:

Jika elemen yang menjadi *independent* berubah, maka elemen yang *dependent* mungkin juga perlu diubah.

🔗 Simbol dalam UML

- Digambarkan dengan **garis putus-putus (dashed line)**.
- Ujung panah mengarah dari **elemen yang bergantung** ke **elemen yang menjadi acuan**.
  👉 *A* → *B* artinya *A bergantung pada B.*

Contoh notasi:

```
ClassA  - - - - - ->  ClassB
```

Artinya: **ClassA memiliki ketergantungan pada ClassB.**

⚙️ Contoh Kasus Dependency

1. **Antarkelas**
   - Sebuah kelas `OrderProcessor` memanggil metode dari kelas `PaymentService`.
     → `OrderProcessor` *bergantung* pada `PaymentService`.
2. **Antarpaket (Package)**
   - Paket `UI` menggunakan fungsi dari paket `Service`.
     → Paket `UI` *bergantung* pada paket `Service`.
3. **Antarmodel lain**
   - Use case “Login” *bergantung* pada use case “Validate User”.

💡 Pengertian Generalization

Generalization menggambarkan hubungan **“is-a”** — yaitu bahwa **objek dari kelas turunan (subclass)** juga merupakan **objek dari kelas induk (superclass)**.

Artinya:

Kelas **subclass** mewarisi **atribut, operasi, dan hubungan** dari kelas **superclass**, serta dapat menambahkan atau memodifikasi perilaku tersebut.

🖇️ Simbol dalam UML

- Digambarkan dengan **garis solid** dengan **panah segitiga kosong** (open arrowhead) menuju **kelas induk (superclass)**.
- Arah panah: dari **kelas anak ke kelas induk**.

Contoh notasi:

`Dog` ——————▷ `Animal`

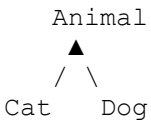
Artinya:
Dog adalah **turunan (subclass)** dari **Animal (superclass)**.
Dengan kata lain, *Dog is an Animal.*

🧱 Contoh dalam Konteks Kelas

Misal kita punya hirarki:

- `Animal` → kelas umum (superclass)
- `Cat`, `Dog` → kelas khusus (subclass)

Maka hubungan UML-nya:



Keterangan:

- `Animal` memiliki atribut: `name`, `age`
- `Cat` dan `Dog` akan **mewarisi** atribut tersebut.
- `Cat` dan `Dog` juga dapat menambahkan atribut/metode baru, misalnya `meow()` atau `bark()`.

💡 Pengertian Realization

Realization berarti *"perwujudan"* atau *"implementasi nyata"* dari suatu spesifikasi.
Dengan kata lain:

Realization digunakan untuk menunjukkan bahwa **kelas konkret (implementor)** *mewujudkan atau mengimplementasikan* perilaku yang dijanjikan oleh **interface atau kontrak (specifier)**.

🖇️ Simbol dalam UL

- Digambarkan dengan **garis putus-putus (dashed line)**
- Ujungnya memiliki **panah segitiga kosong** (open arrowhead)
- Arah panah mengarah dari **kelas pengimplementasi** ke **interface atau spesifikasi**

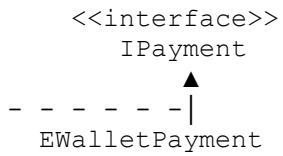
Contoh notasi:

```
PaymentService ┄ - - -▷ IPayment
```

Artinya:
PaymentService *mengimplementasikan* **IPayment**.

🧱 Contoh dalam Konteks Kelas dan Interface

Diagram UML-nya:



Artinya:

- `IPayment` mendefinisikan kontrak (spesifikasi).
- `EWalletPayment` mewujudkan (merealisasikan) kontrak tersebut dengan implementasi konkret.

🧩 Contoh dalam Use Case

Dalam use case diagram, *realization* juga digunakan untuk menunjukkan bahwa **komponen atau kelas** *merealisasikan* **use case tertentu**.

Contoh:

```
Use Case: "Proses Pembayaran"
direalisasikan oleh → Class "PaymentProcessor"
```

💡 Pengertian Association

Association adalah **relasi yang menunjukkan bahwa suatu objek dari satu kelas berhubungan atau berinteraksi dengan objek dari kelas lain**.
Artinya:

Objek dari kelas A *berasosiasi* dengan objek dari kelas B untuk melakukan suatu fungsi atau bertukar informasi.

Dengan kata lain, **association menggambarkan adanya "hubungan memiliki atau menggunakan" antar objek.**

🔗 Simbol dalam UML

- Digambarkan dengan **garis solid (garis lurus)** yang **menghubungkan dua kelas**.
- Kadang dilengkapi dengan:
  - **Nama relasi**
  - **Arah navigasi (panah)**
  - **Multiplicities (kardinalitas)** di tiap ujung (misalnya `1`, `0..*`, `1..*`, dll)
  - **Peran (role)** di masing-masing sisi.

Contoh notasi:

```
Customer ——— Order
```

Atau lebih lengkap:

```
Customer 1 ——— * Order
```

Artinya:
Satu **Customer** dapat memiliki banyak **Order** (*one-to-many association*).

🧱 Contoh dalam Konteks Kelas

Contoh:

- Kelas `Dosen` dan `Mahasiswa`
  Dosen membimbing beberapa Mahasiswa.

Maka relasinya:

```
Dosen 1 ——— * Mahasiswa
```

Artinya:

- Satu **Dosen** membimbing banyak **Mahasiswa**
- Tapi satu **Mahasiswa** hanya dibimbing oleh satu **Dosen**

🧩 Jenis-Jenis Association

1. **Binary Association**
   → Hubungan antara dua kelas (yang paling umum).
   Contoh: `Customer` — `Order`.
2. **Ternary Association**
   → Hubungan antara tiga kelas.
   Contoh: `Dokter` — `Pasien` — `Obat`.
3. **Reflexive Association**
   → Sebuah kelas berasosiasi dengan dirinya sendiri.
   Contoh: `Karyawan` dapat menjadi `atasan` bagi `Karyawan` lain.
4. **Qualified Association**
   → Memiliki "kunci pengenal" untuk memperjelas hubungan.
   Contoh: `Order` diidentifikasi dengan `orderNumber`.

🧬 Special Forms of Association

1. **Aggregation (Has-a relationship)**
   - Hubungan "bagian dari" (part-whole) tetapi **tidak mutlak**.
   - Digambarkan dengan **belah ketupat putih** di sisi yang "memiliki".
     Contoh: `Perpustakaan` *memiliki* `Buku`.
2. **Composition (Strong ownership)**
   - Hubungan "bagian dari" yang **tidak bisa dipisahkan**.
   - Digambarkan dengan **belah ketupat hitam** di sisi pemilik.
     Contoh: `Mobil` *terdiri dari* `Mesin` (jika mobil dihapus, mesin juga hilang).