

Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak  
**OO Analysis and Design**

SEMESTER II TAHUN AJARAN 2022/2023



KNOWLEDGE & SOFTWARE ENGINEERING

# *Contoh Pembuatan Diagram Kelas*



# *Urutan Langkah*

- Analisa kebutuhan dari Pengguna
  - Amati 'Kata Benda' → kandidat kelas
  - Amati 'Kata kerja' yang terkait kata benda → kandidat asosiasi antar kelas, atau kandidat operasi dalam kelas
  - Gambar Diagram Kelas
    - Tentukan atribut
  - Gambarkan Diagram Objek untuk membantuk menjelaskan diagram Kelas

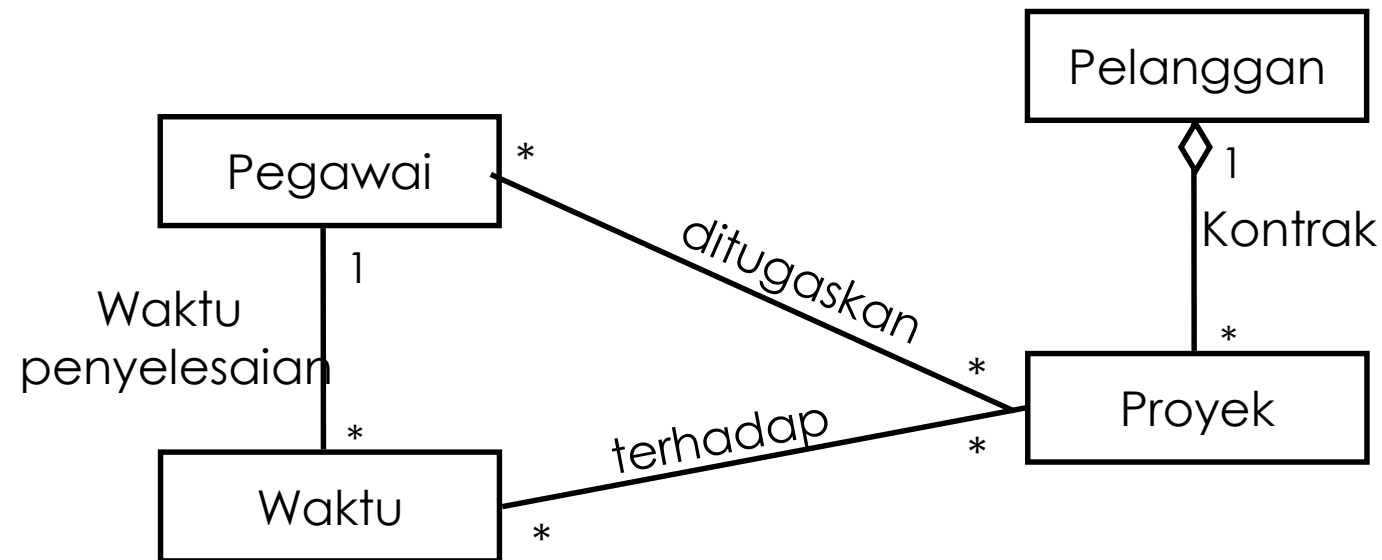
## ***Contoh***

- Kita diminta mengembangkan sistem untuk mencatat waktu para pegawai untuk menyelesaikan suatu proyek dari pelanggan/user
- Proyek akan dibagi menjadi sekumpulan aktivitas, dan aktivitas akan dibagi menjadi sekumpulan pekerjaan. Setiap pekerjaan akan ditugaskan kepada seorang pegawai yang bekerja full time atau part-time
- Setiap pekerjaan membutuhkan keahlian tertentu dan pegawai memiliki keahlian yang berbeda dengan tingkat pengalaman yang berbeda



## Contoh

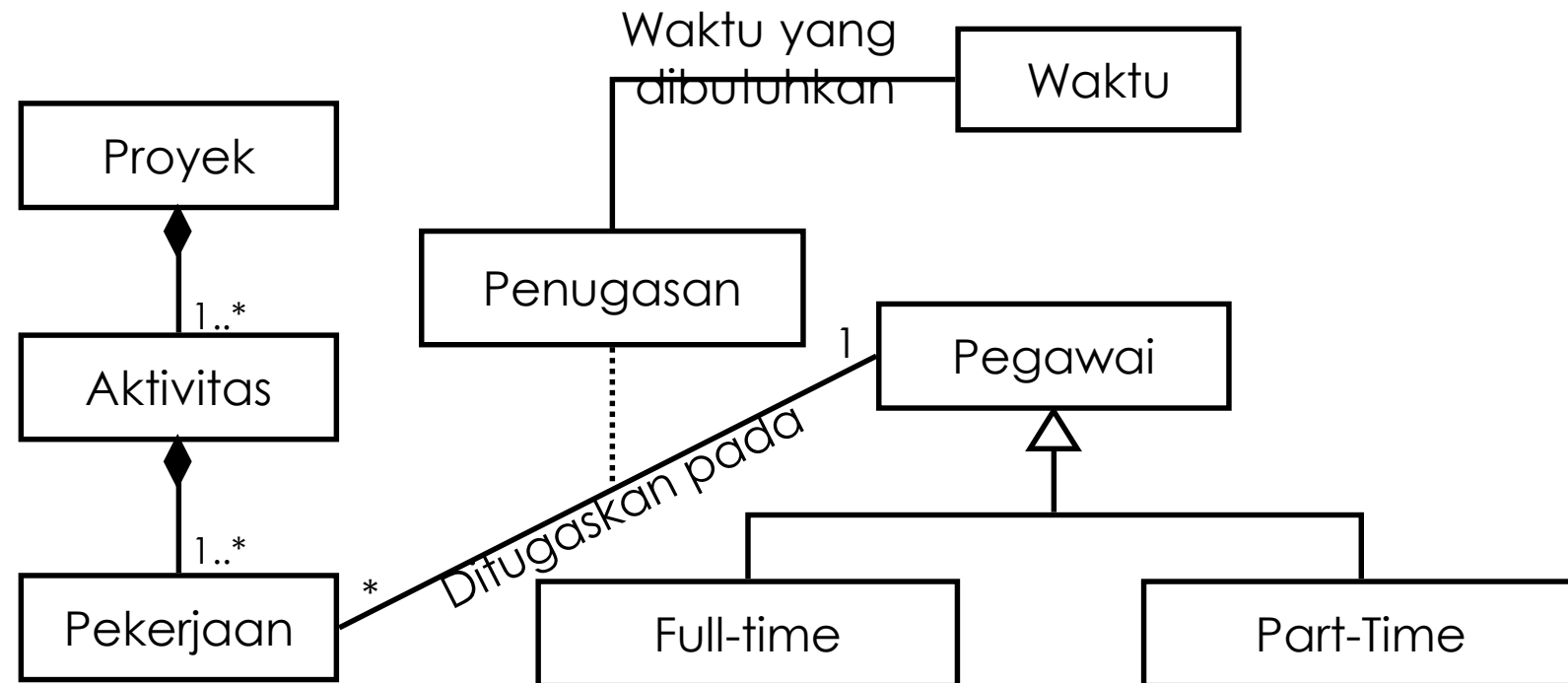
- Kita diminta mengembangkan sistem untuk mencatat waktu para pegawai untuk menyelesaikan suatu proyek dari pelanggan/user



# Contoh

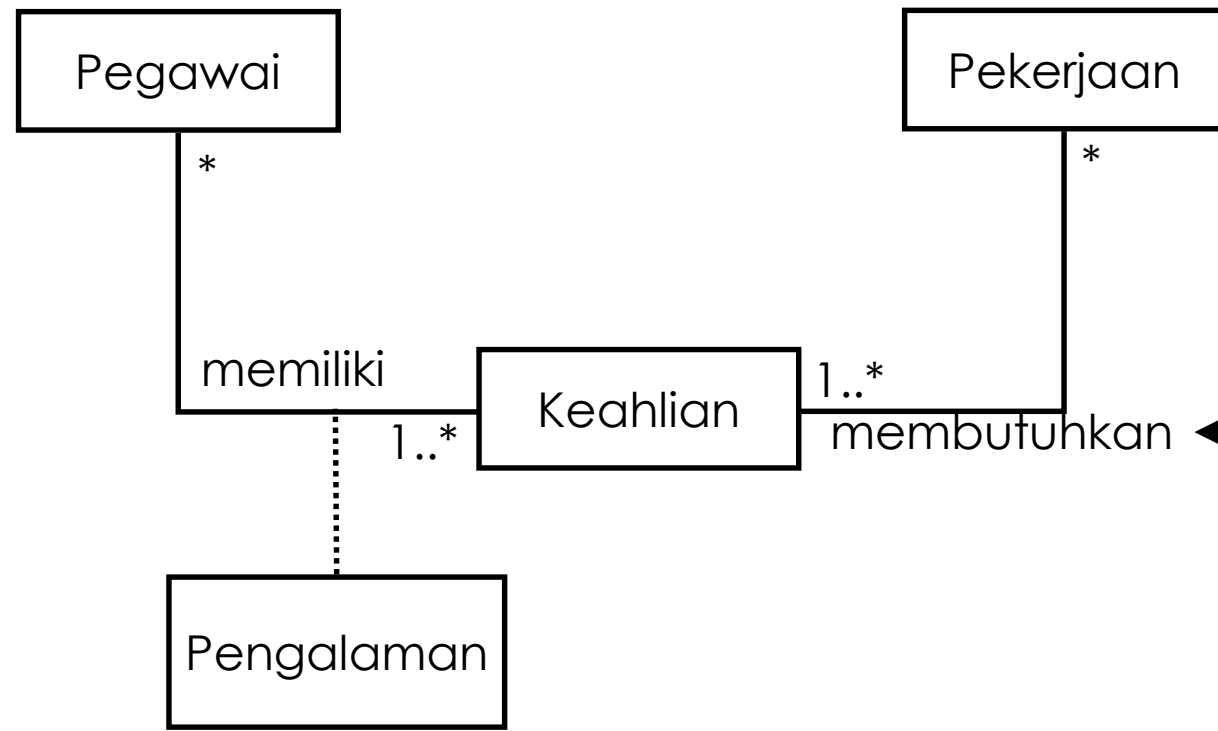
6

- Proyek akan dibagi menjadi sekumpulan aktivitas, dan aktivitas akan dibagi menjadi sekumpulan pekerjaan. Setiap pekerjaan akan ditugaskan seorang pegawai, yang bekerja full time atau part-time



# Contoh

- Setiap pekerjaan membutuhkan keahlian tertentu dan pegawai memiliki keahlian yang berbeda dengan tingkat pengalaman yang berbeda



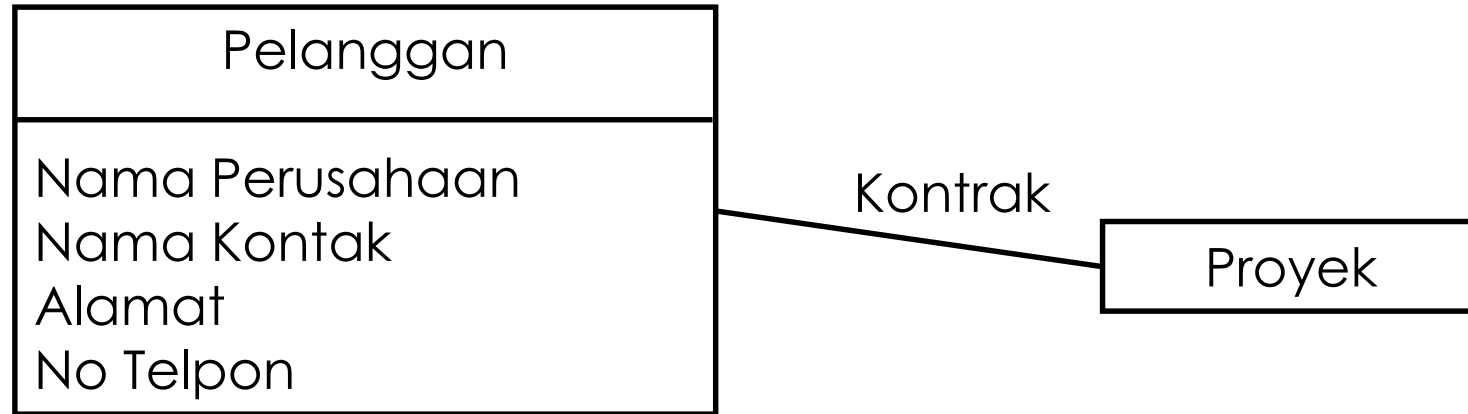
# *Urutan Langkah*

- Analisa kebutuhan dari Pengguna
  - Amati 'Kata Benda' -> kandidat kelas
  - Amati 'Kata kerja' yang terkait kata benda → kandidat asosiasi antar kelas, atau kandidat operasi dalam kelas
  - **Gambar Diagram Kelas**
    - **Tentukan atribut**
    - Gambarkan Diagram objek untuk membantuk menjelaskan diagram Kelas
- Cari 'Use Case'
  - Identifikasi Aktor
  - Identifikasi Use Case
  - Cari hubungan antara Aktor, juga antara use-case
  - Gambarkan diagram Use-Case
- Gambarkan Diagram Sekuens (Sequence Diagram)
  - Satu 'Use-Case' memiliki satu diagram sekeuns
  - Pada diagram sekuens akan memperlihatkan tugas suatu kelas
- Tambahkan Metode/Operasi pada Diagram Kelas





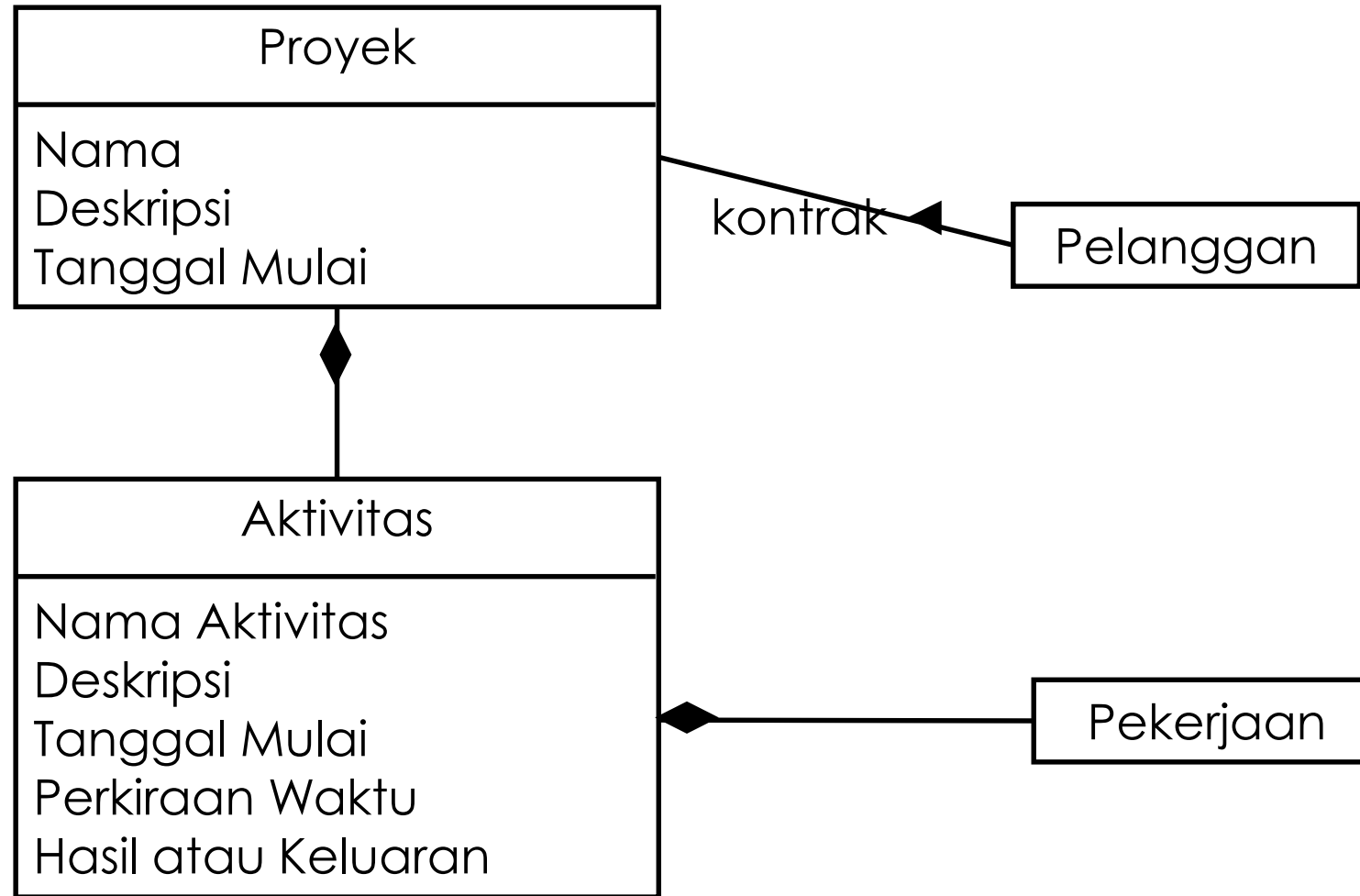
# Contoh



## Catatan

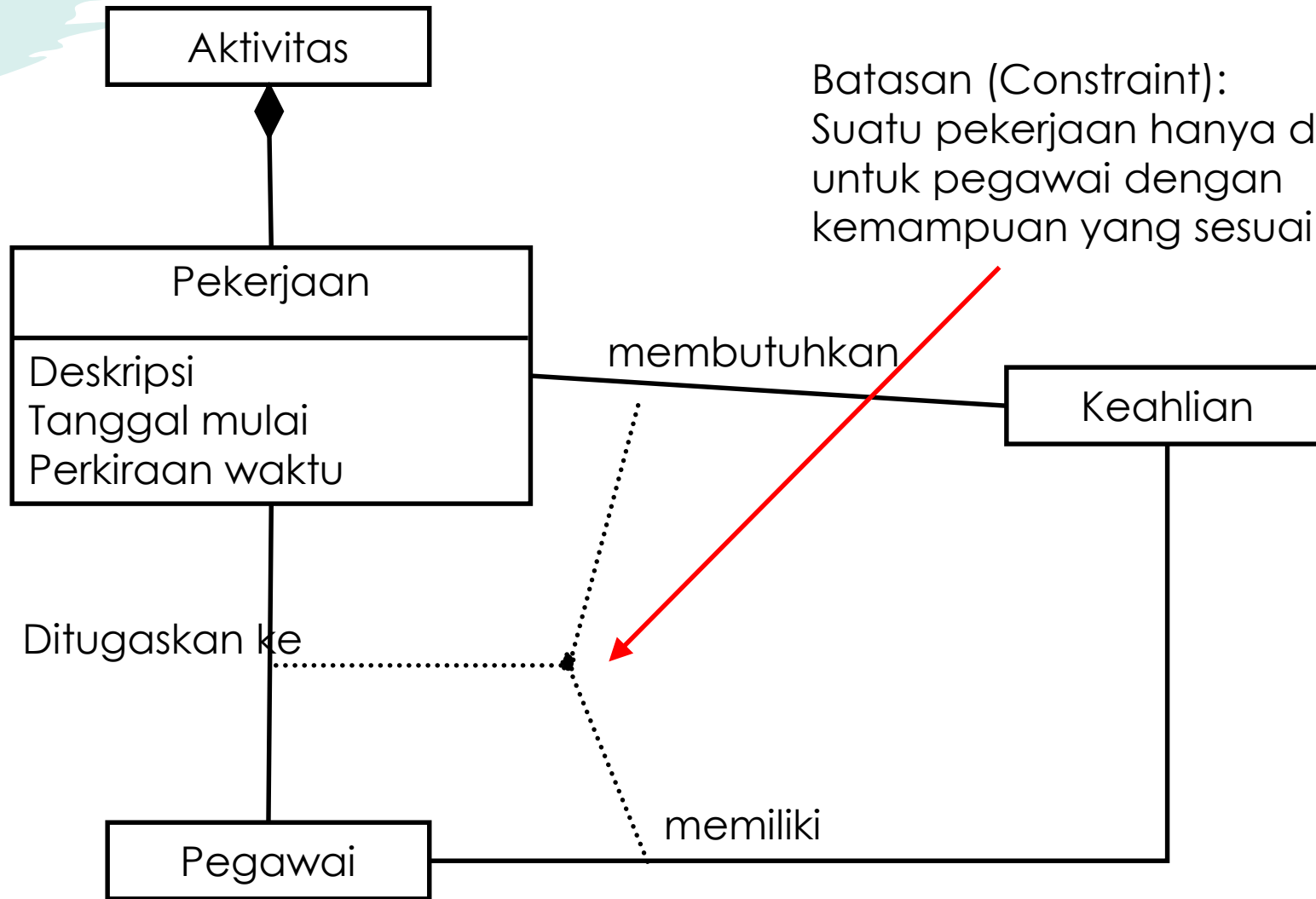
- Proyek tidak memiliki atribut terkait pelanggan
  - maka digunakan asosiasi 'Kontrak'
- Mungkin tidak perlu menuliskan nomor id pelanggan
  - Dalam analisis, hanya menampilkan yang perlu saja
  - Nomor id mungkin hanya akan diperhatikan saat perancangan

# Contoh



# Contoh

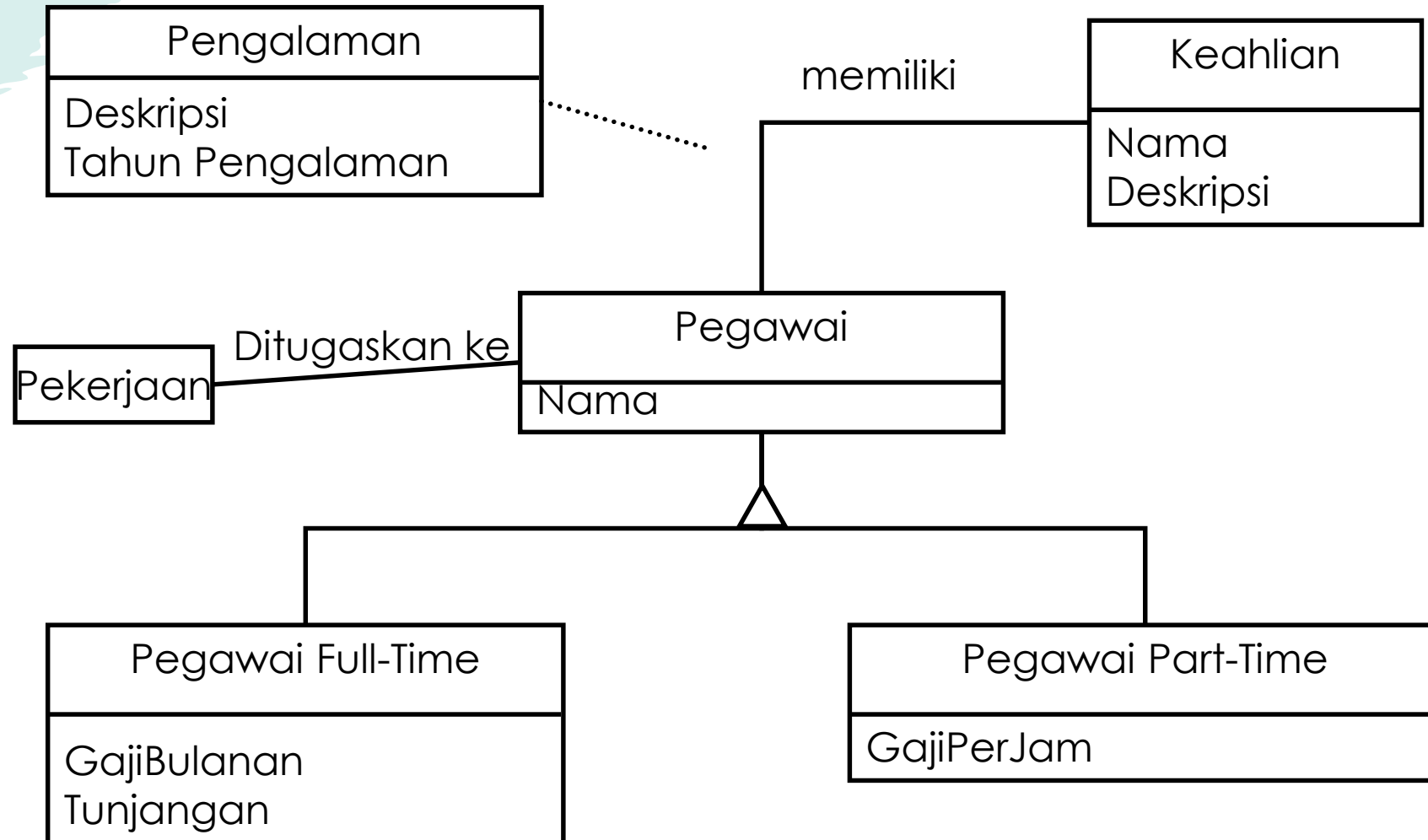
11



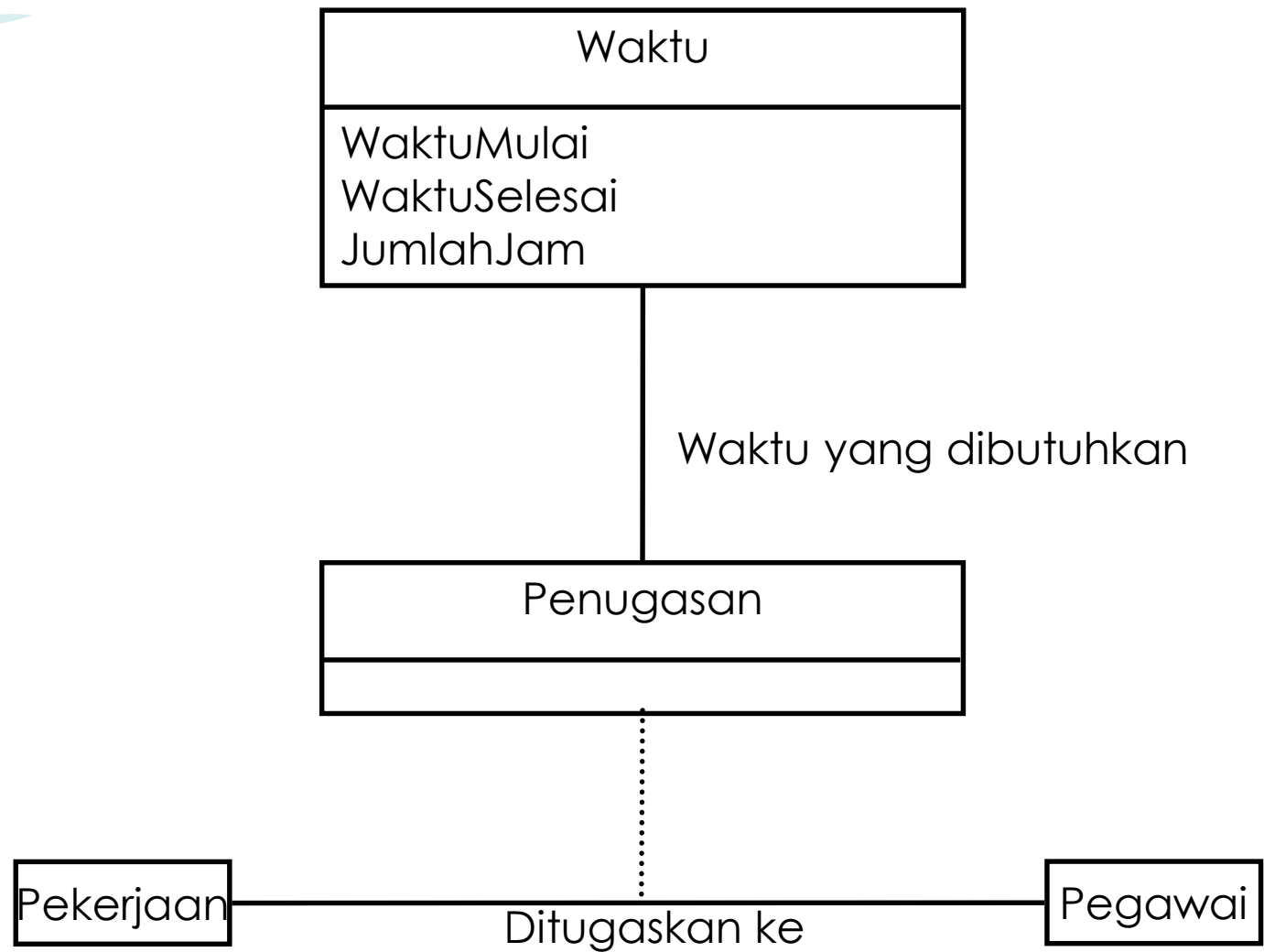
Batasan (Constraint):  
Suatu pekerjaan hanya diberikan  
untuk pegawai dengan  
kemampuan yang sesuai

# Contoh

12



# Contoh

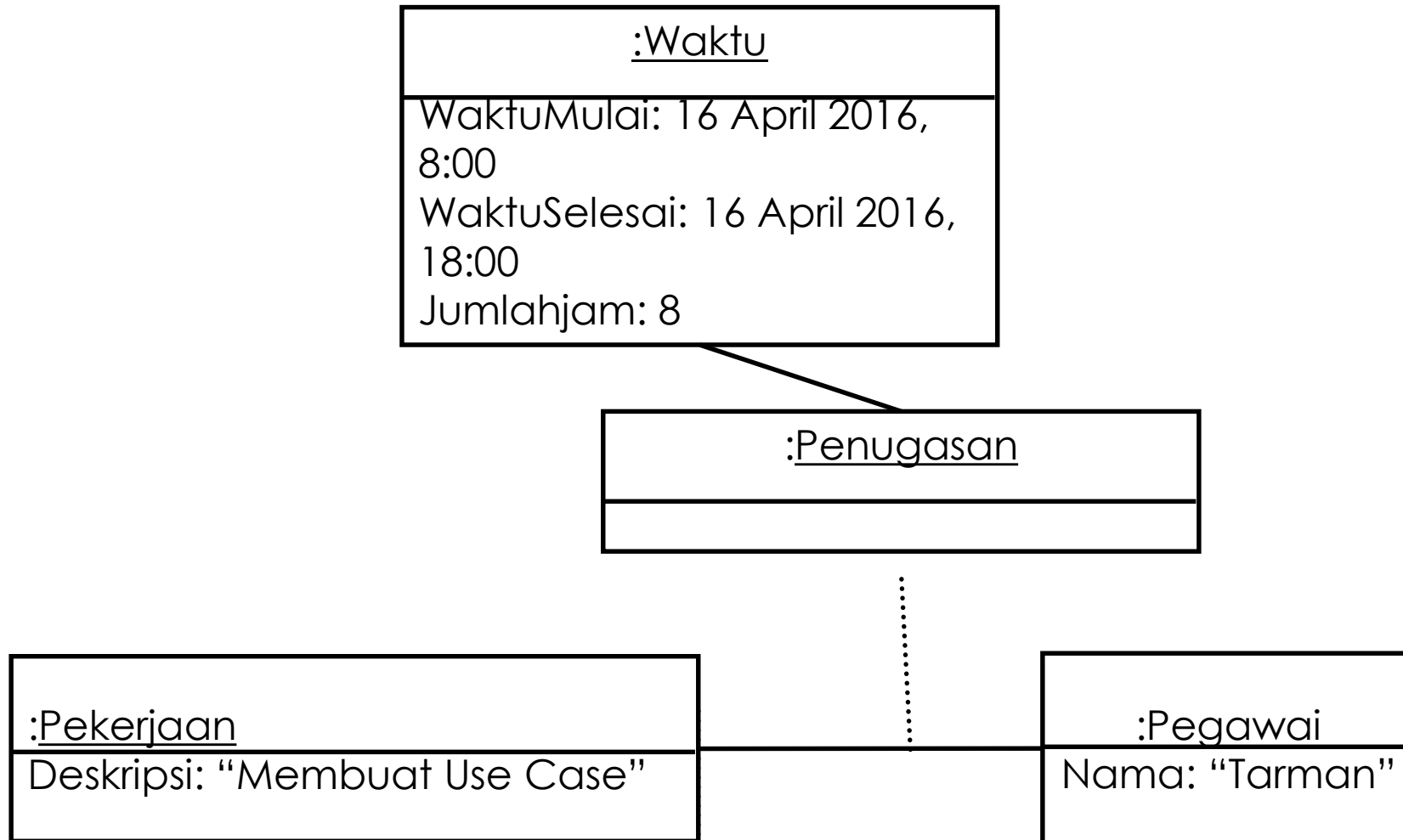


# *Urutan Langkah*

- Analisa kebutuhan dari Pengguna
  - Amati 'Kata Benda' -> kandidat kelas
  - Amati 'Kata kerja' yang terkait kata benda → kandidat asosiasi antar kelas, atau kandidat operasi dalam kelas
  - Gambar Diagram Kelas
    - Tentukan atribut
  - **Gambarkan Diagram objek untuk membantu menjelaskan diagram Kelas**



# *Diagram objek*



# ***Contoh Pembuatan Diagram Use Case***





# *Urutan Langkah*

- **Cari 'Use Case'**

- Identifikasi Aktor
- Identifikasi Use Case
- Cari hubungan antara Aktor, juga antara use-case
- Gambarkan diagram Use-Case

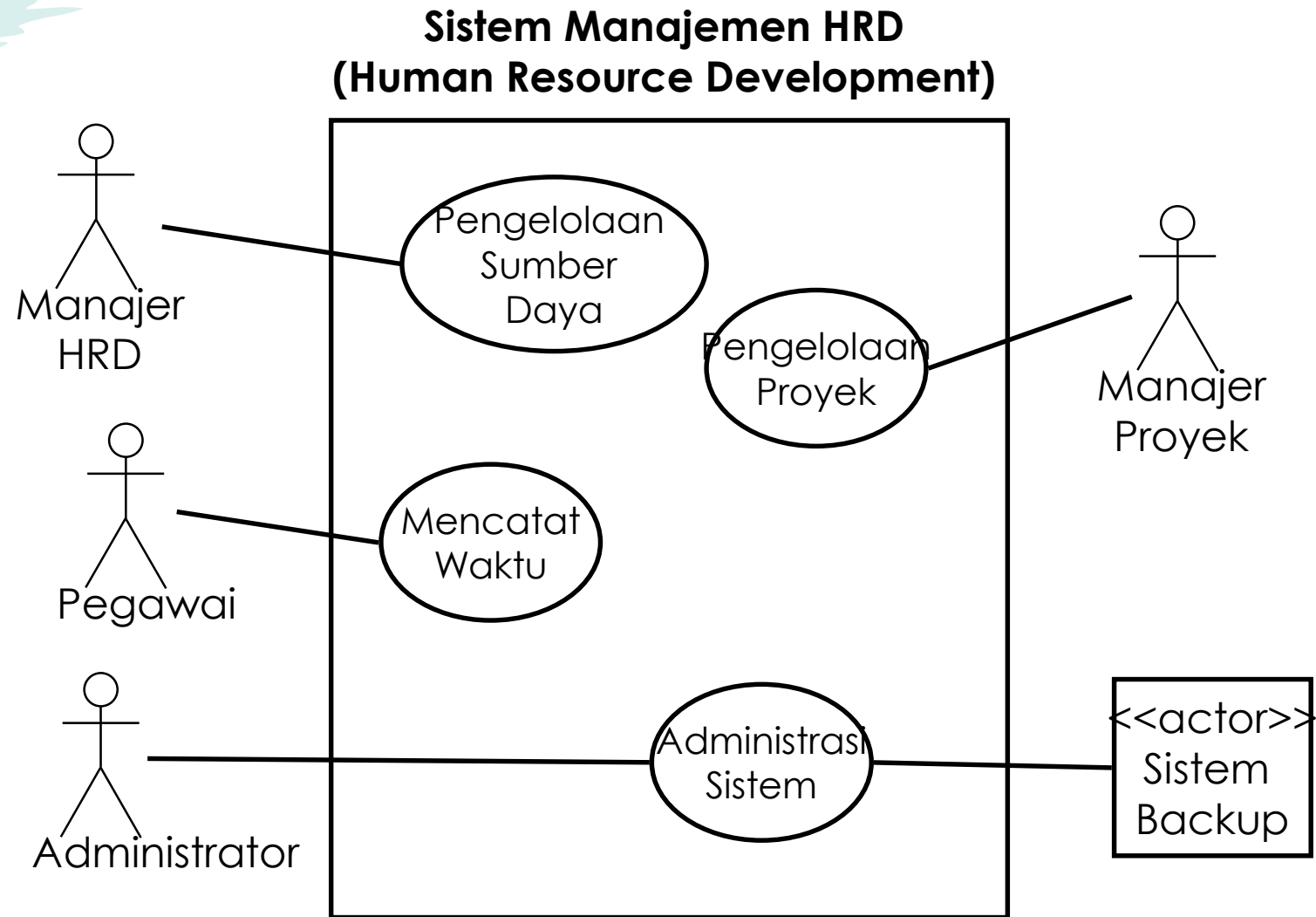


# *Use Case*

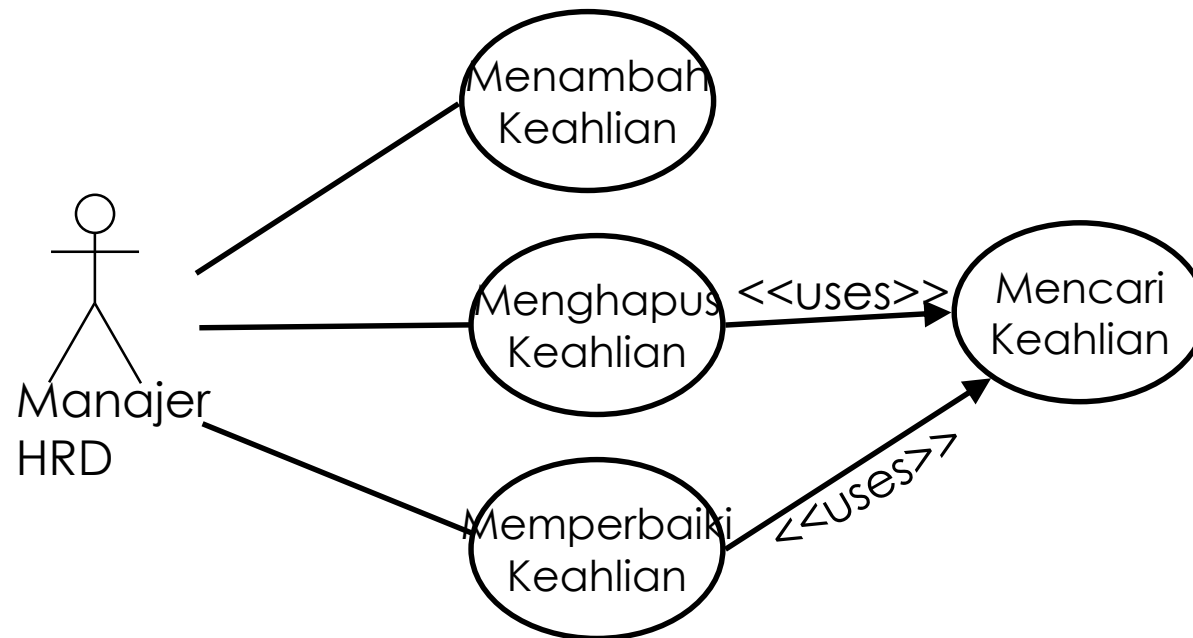
- Aktor:
  - Mewakili pengguna dari sistem
    - Pengguna (manusia)
    - Sistem lain
- Use cases
  - Use case mewakili suatu fungsionalitas atau layanan yang diberikan sistem untuk aktor



# Diagram Use Case



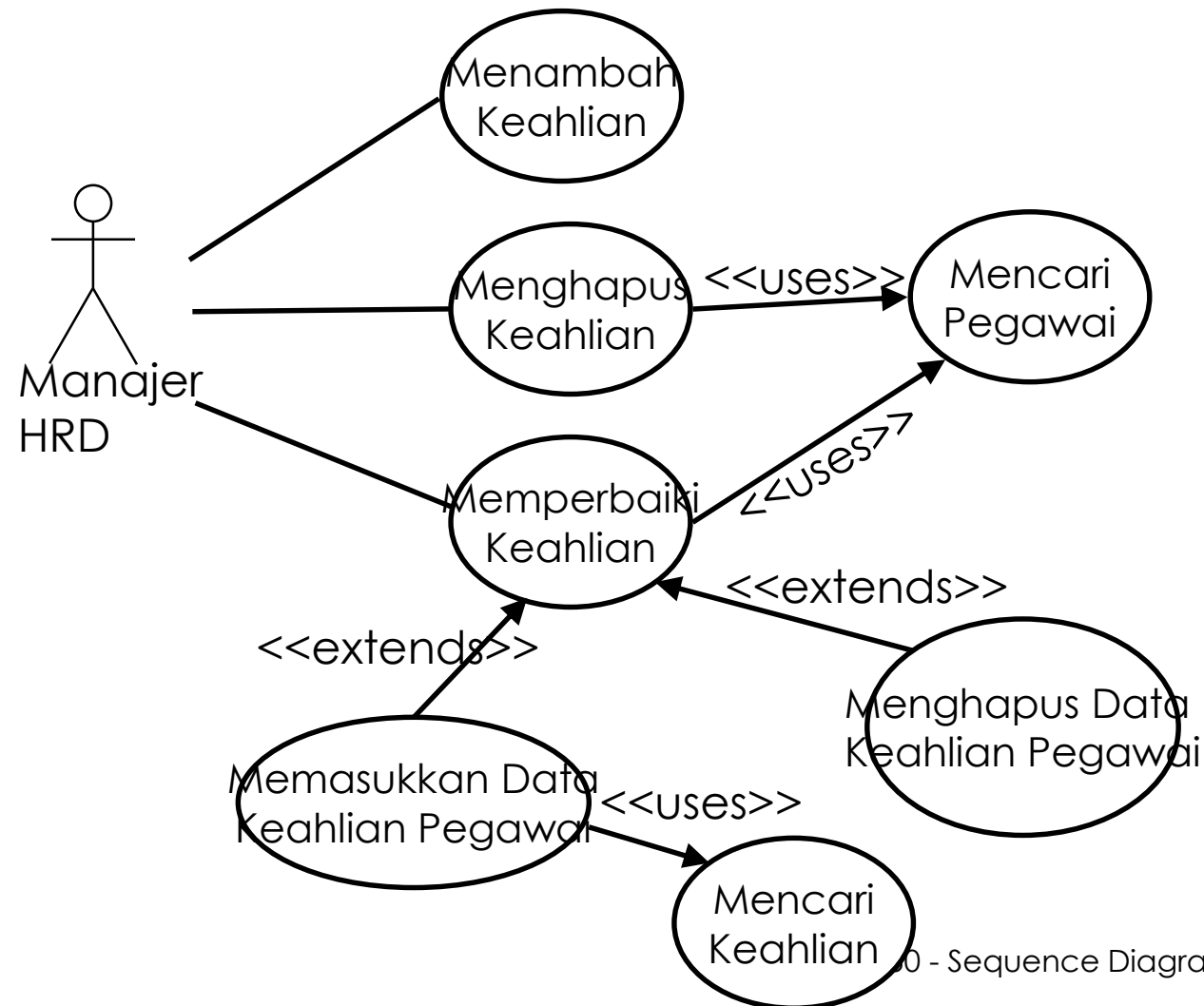
# *Use Case Pengelolaan Sumber Daya*



Perhatikan bahwa contoh yang diberikan ini adalah use-case untuk sistem yang relatif sederhana. Dalam system yang lebih besar, kerinciannya mungkin tidak sampai ke proses **CRUD** (seperti pada contoh di atas)

# ***Use Case Pengelolaan Sumber Daya (Perbaikan)***

21



10 - Sequence Diagram

# *Sequence diagram*



# ***Pendahuluan***

- Diagram ini digunakan sebagai penjelasan dari deskripsi use-case
  - Diagram ini menampilkan bagaimana suatu event menyebabkan transisi antar objek
  - Menampilkan scenario eksekusi dalam sistem
- **Aktor** dan **kelas-kelas** yang terlibat ditampilkan di bagian atas
- Aktivasi dari aktor dan objek akan ditampilkan sebagai segiempat vertikal sepanjang garis putus-putus yang disebut **lifelines**
- **Panah** penghubung aktivasi diberi nama **event** tsb.  
Panah ini menunjukkan arah transisi dari aktor ke objek atau antar objek
- Aliran antar objek direpresentasikan dengan suatu label pada garis putus-putus horizontal dengan nama objek yang dikirim
- Status (states) dapat ditunjukkan pada lifelines



# ***Elemen Diagram Sekuens***

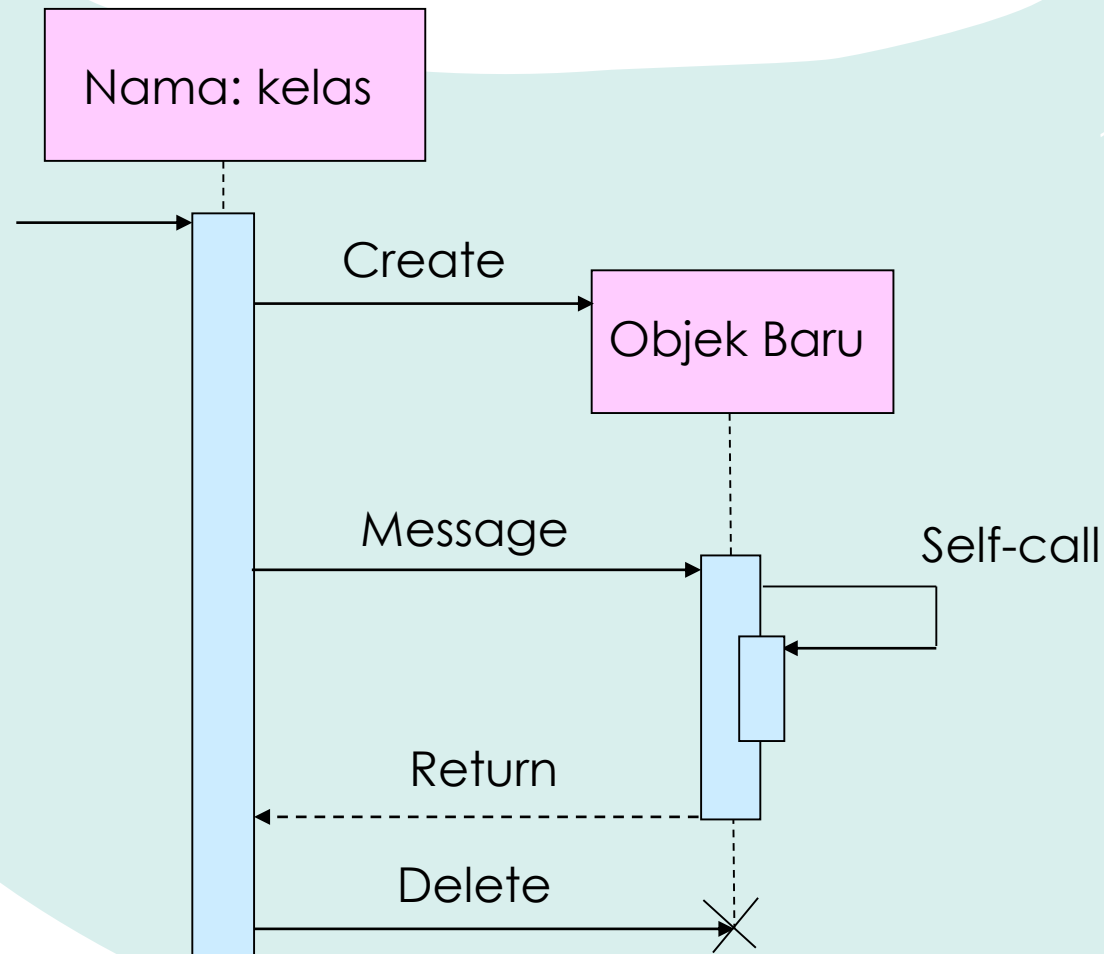
- **Peserta (Participant)/aktor:** suatu objek yang diperlukan pada suatu diagram sekuens
- **Pesan (Message):** komunikasi antar objek peserta
- Garis vertical arah bawah menunjukkan urutan waktu eksekusi
  - dimulai dari atas, diteruskan ke bawahnya





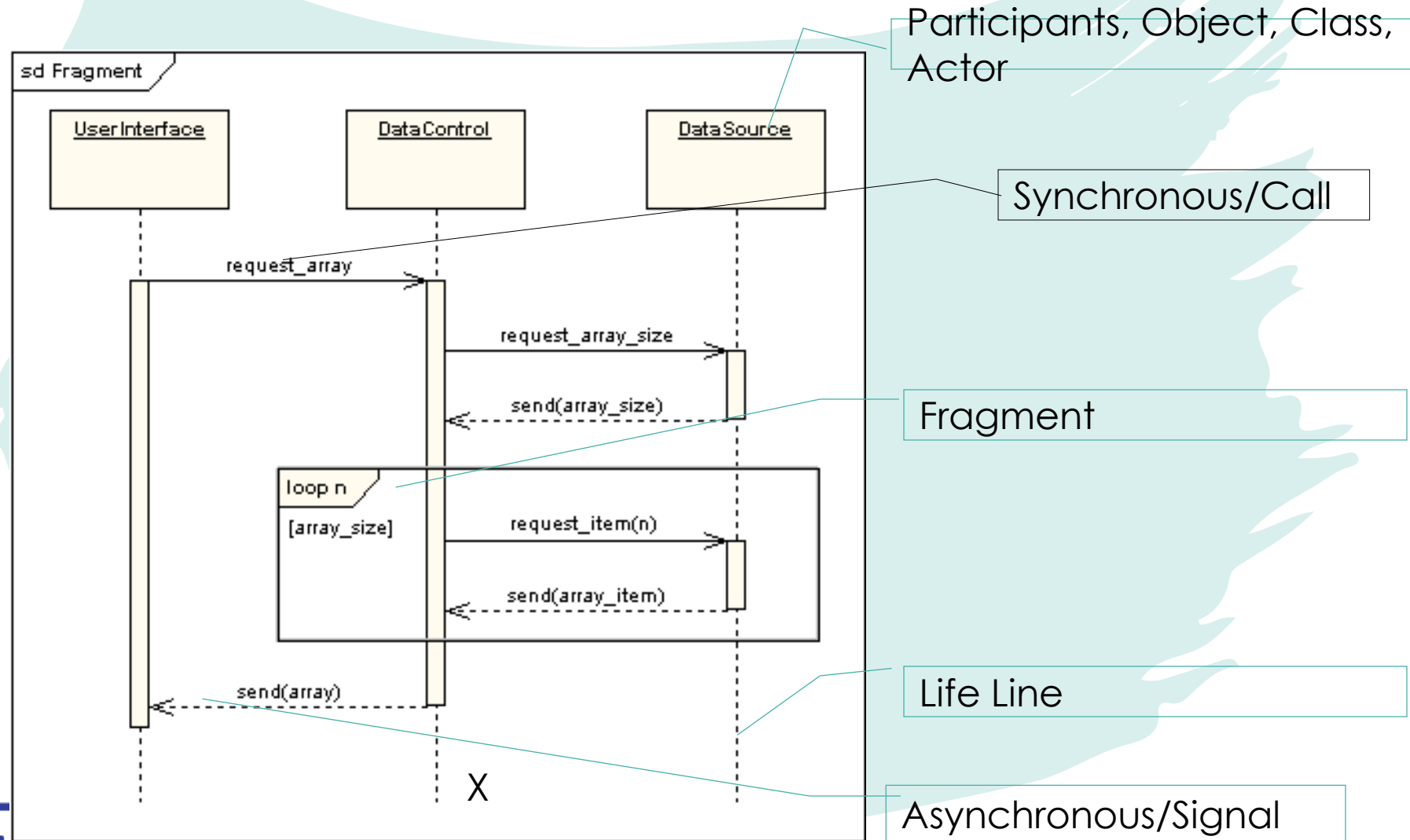
# *Elemen pada Diagram Sekuens*

25

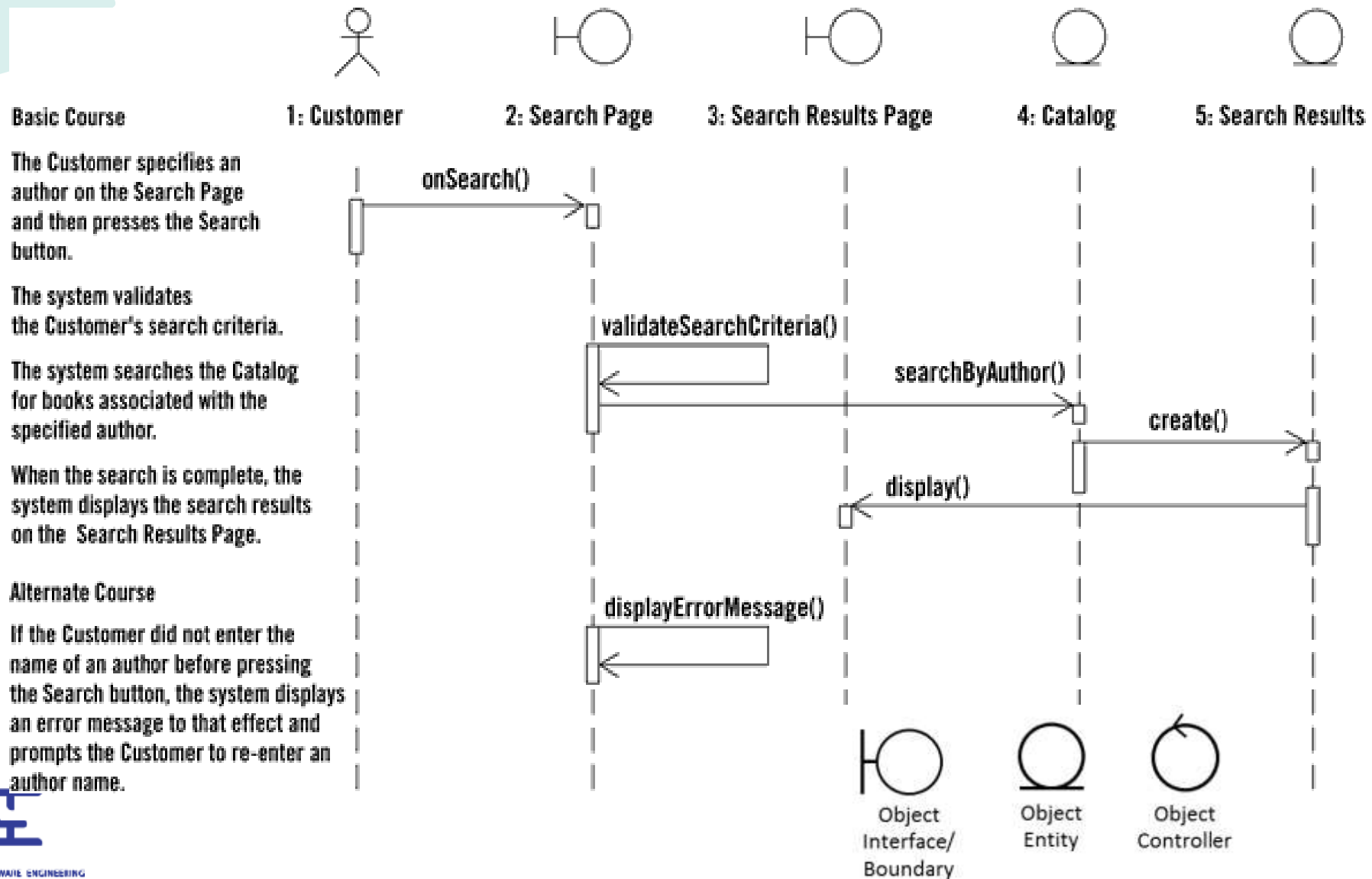


Gambar tsb adalah representasi pemanggilan suatu operasi dengan pengiriman pesan

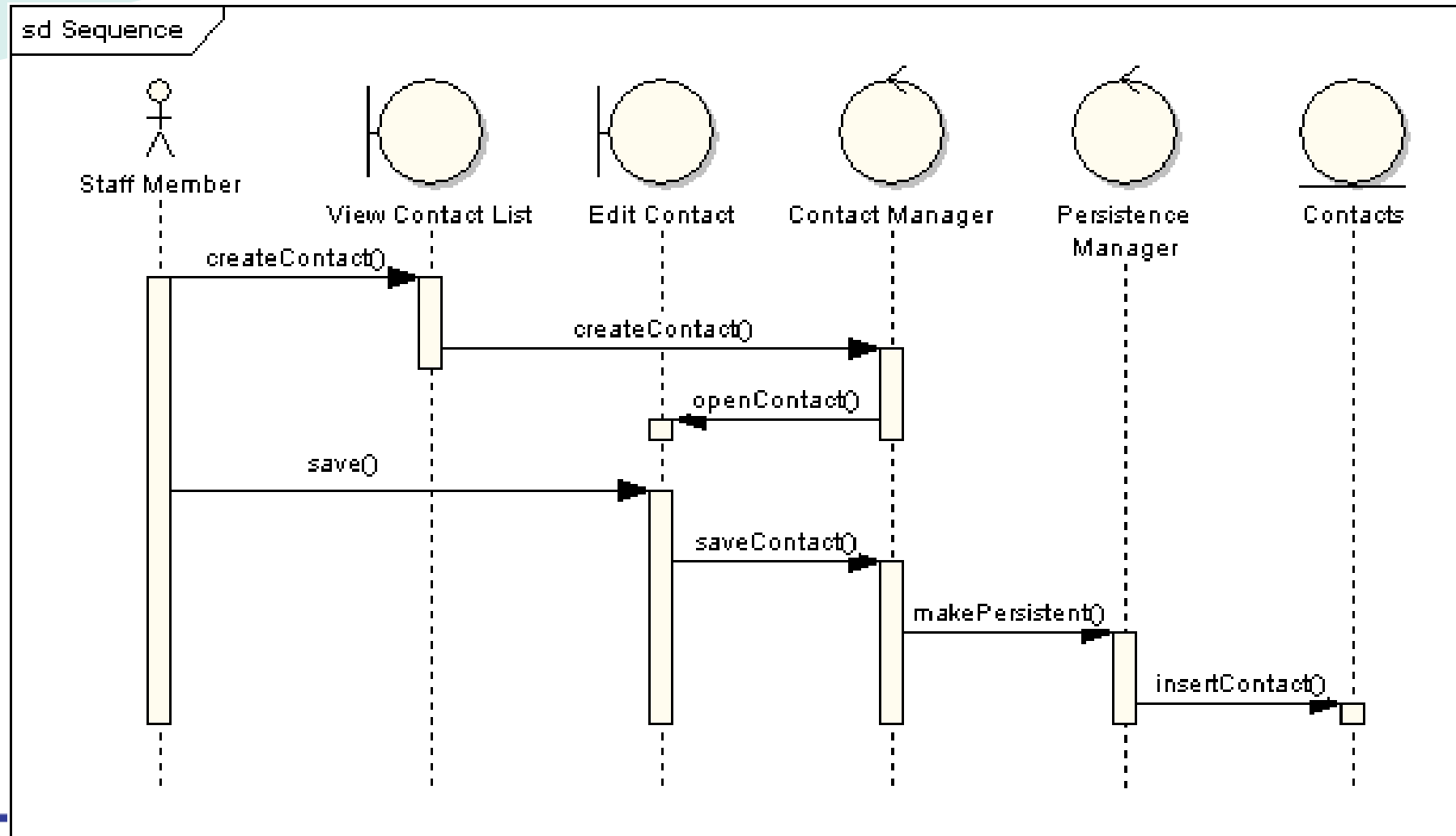
# Sequence Diagram



# Diagram Sekuens dari use case

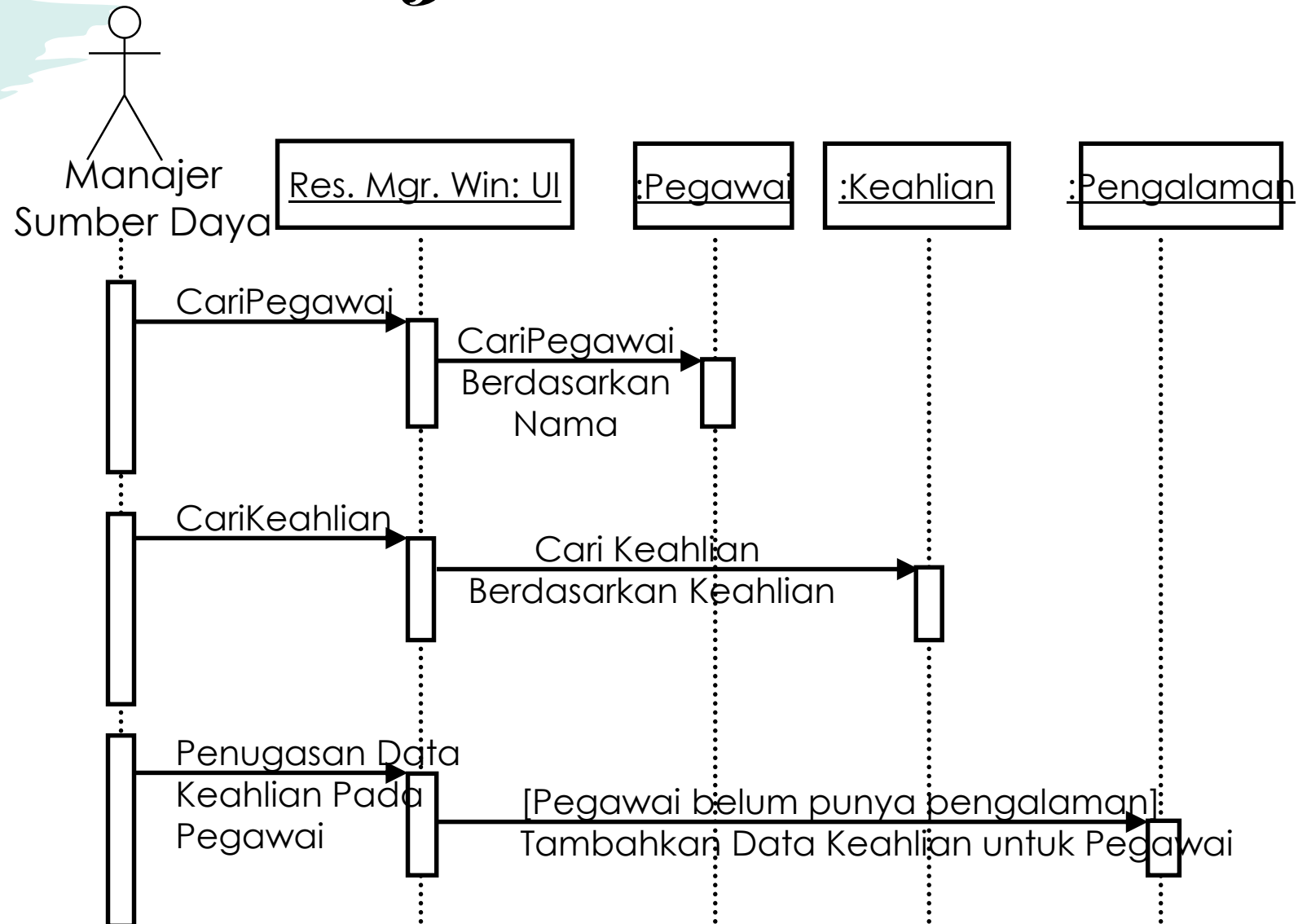


# Sequence Diagram



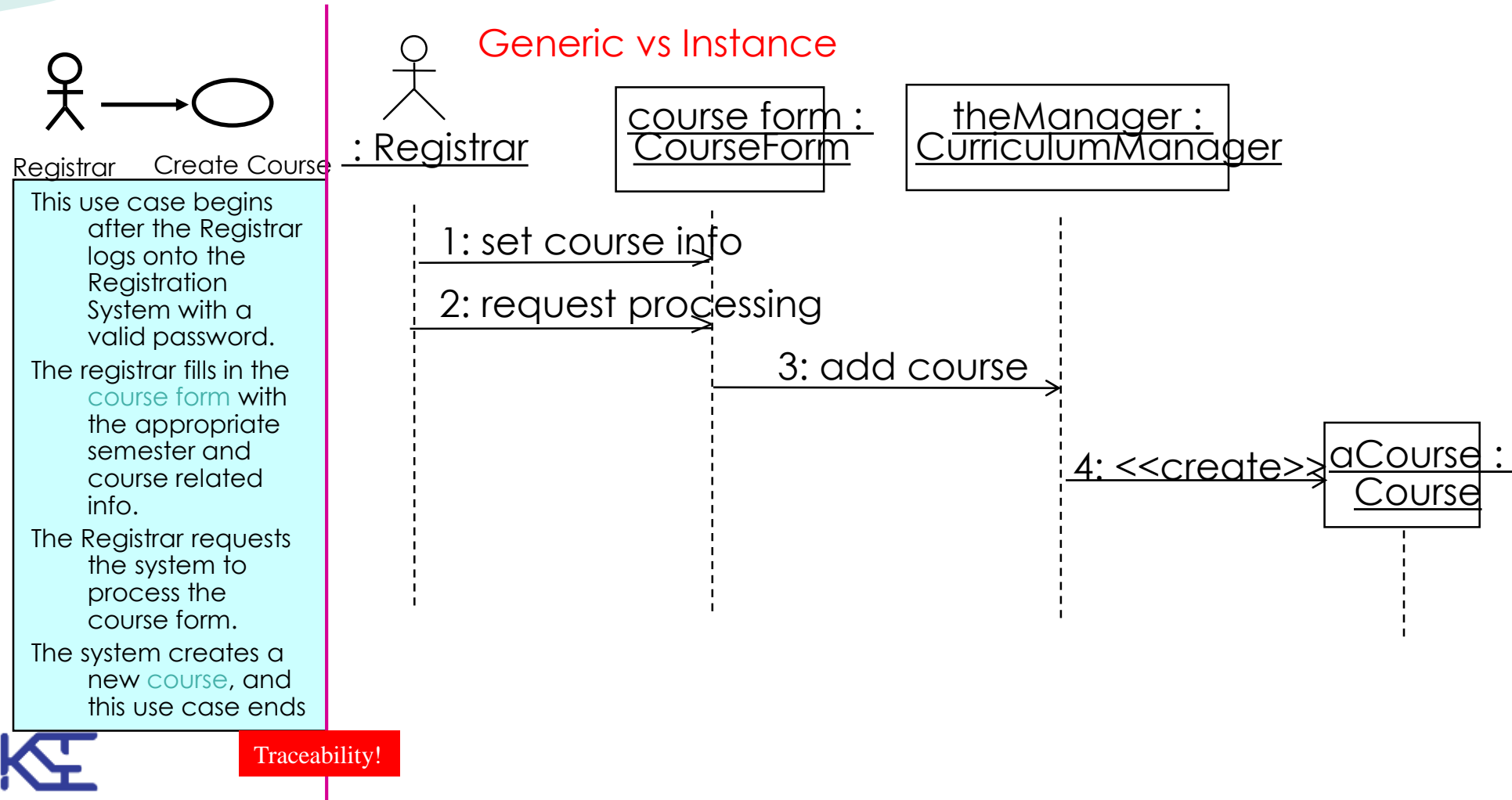
# Contoh Diagram Sekuens

29



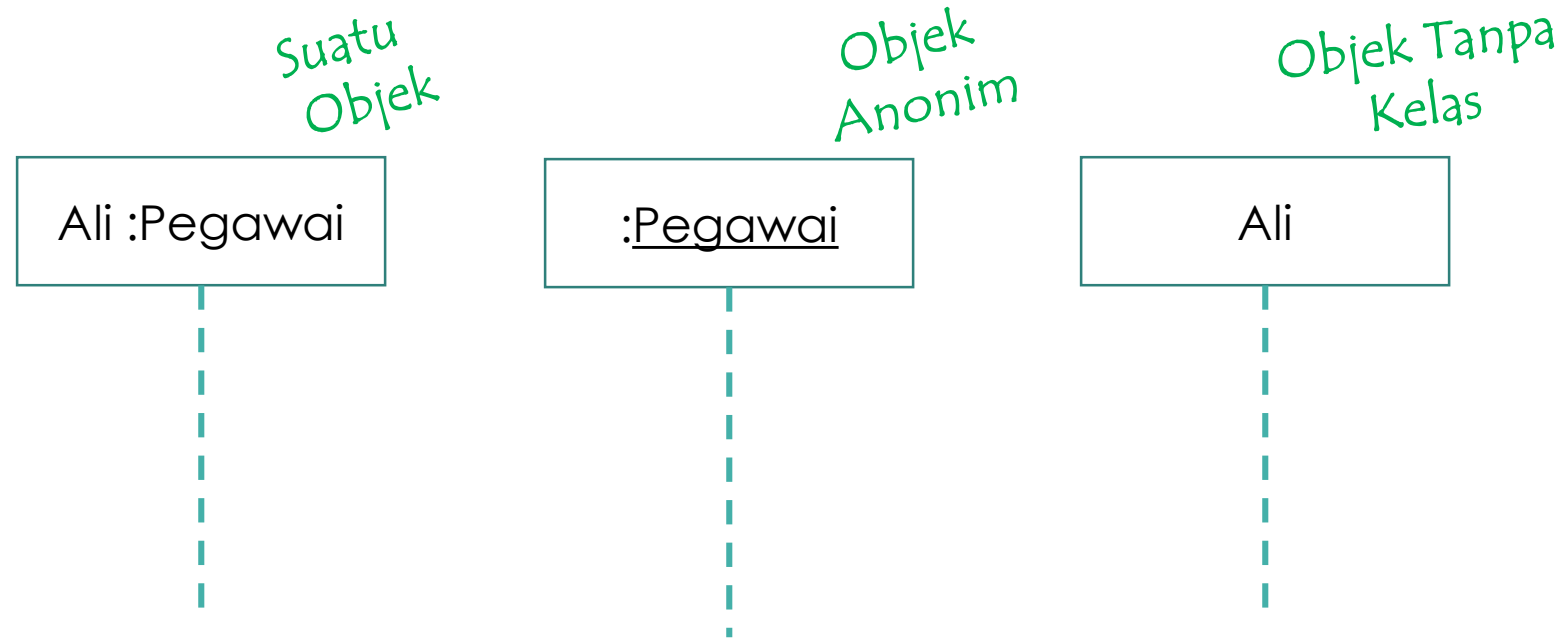
# Sequence Diagram

- Sequence diagram menampilkan **interaksi objek** yang disusun dalam **urutan waktu**



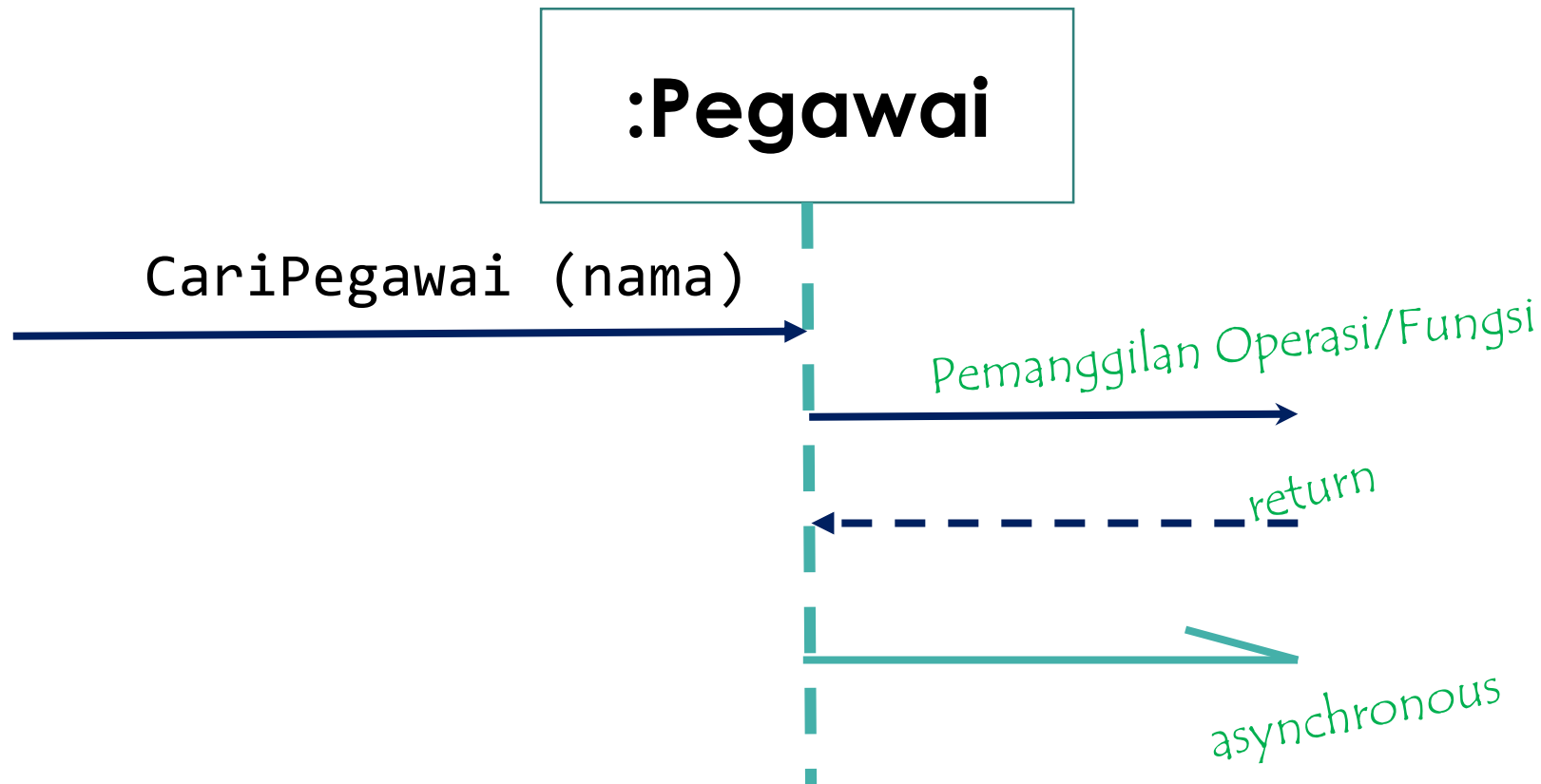
# *Penamaan Objek*

- Penamaan objek dimulai dengan titik dua, lalu nama objek
  - Jika perlu, berikan nama objek untuk memperjelas
  - Garis dibuat terputus-putus
  - Aturan penulisan: <nama objek> :<nama kelas>



# ***Pesan (Message) antar Objek***

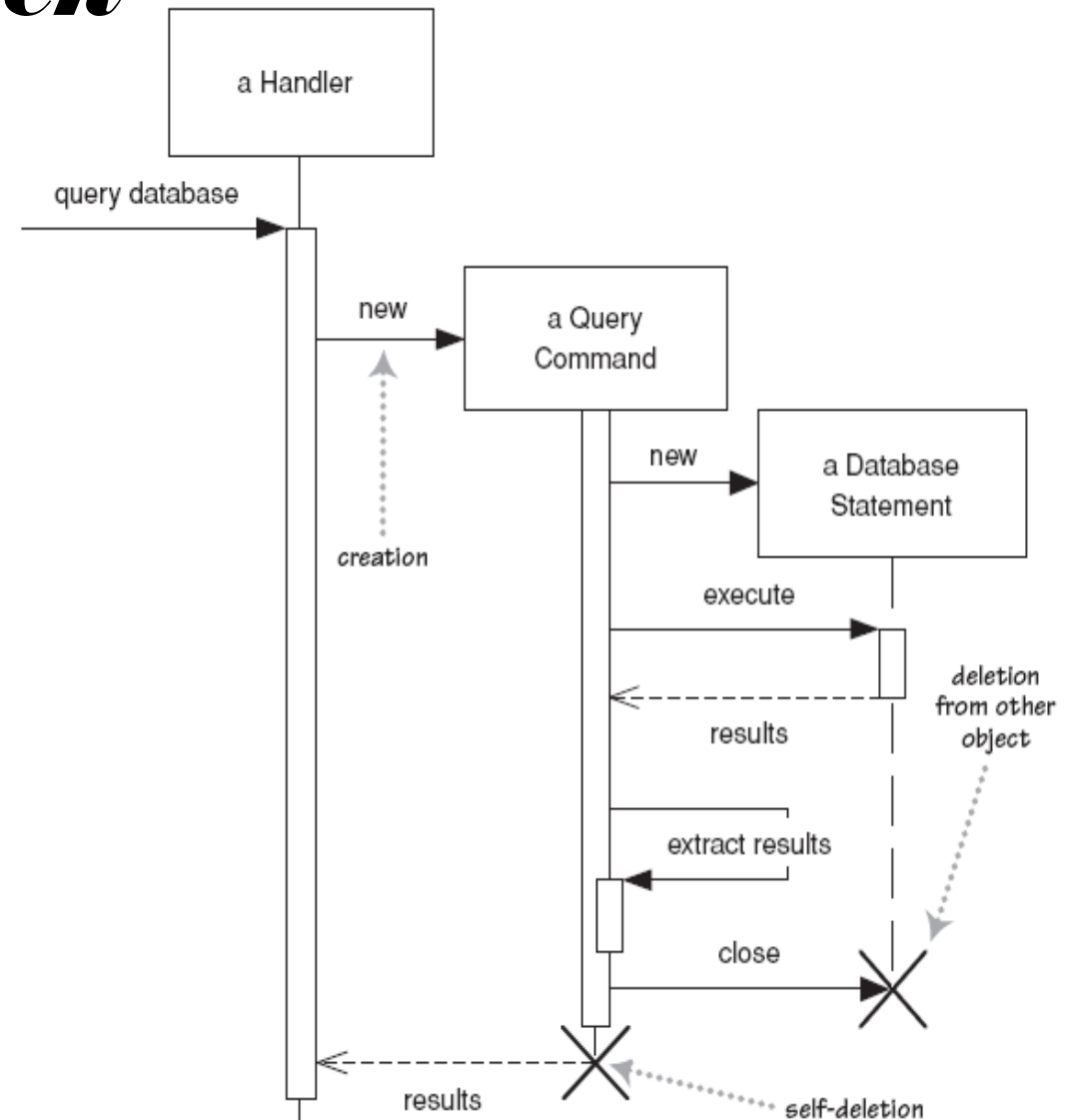
- Pesan digambarkan sebagai garis horizontal dengan arah panah
  - Nama pesan dan argumennya ditulis di atas garis panah





# ***Daur hidup suatu Objek***

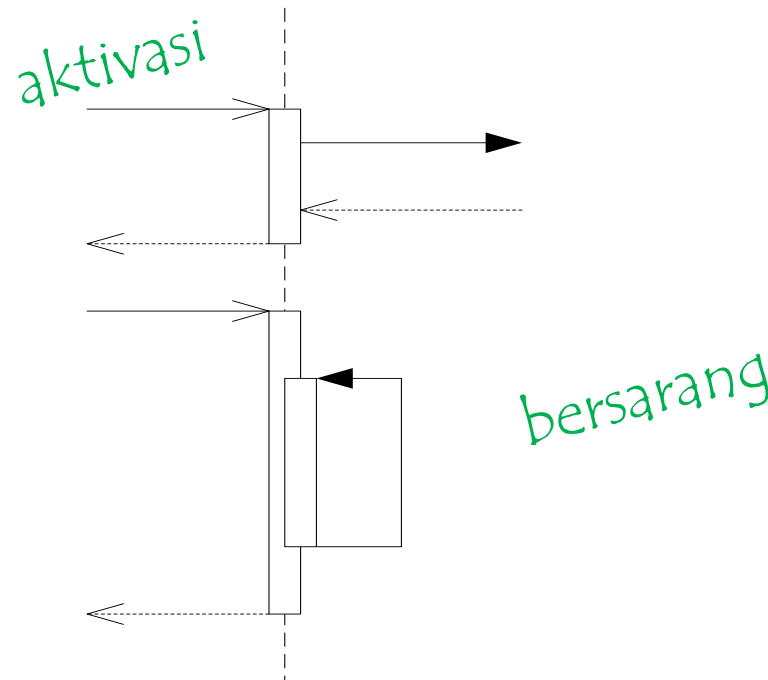
- Konstruksi objek (*creation*)
  - Panah dengan tulisan 'new'
  - Perhatikan objek yang dibuat setelah awal suatu scenario akan dituliskan lebih rendah
- Destruksi objek (*deletion*):
  - X besar di bawah garis horizontal menandai penghapusan objek
  - Pada Bahasa Java, objek tidak secara eksplisit dihapus, setelah keluar dari scope, maka akan ditangani oleh system 'garbage collection'



IF2250 - Sequence Diagram

# *Pemanggilan suatu Metode/Operasi*

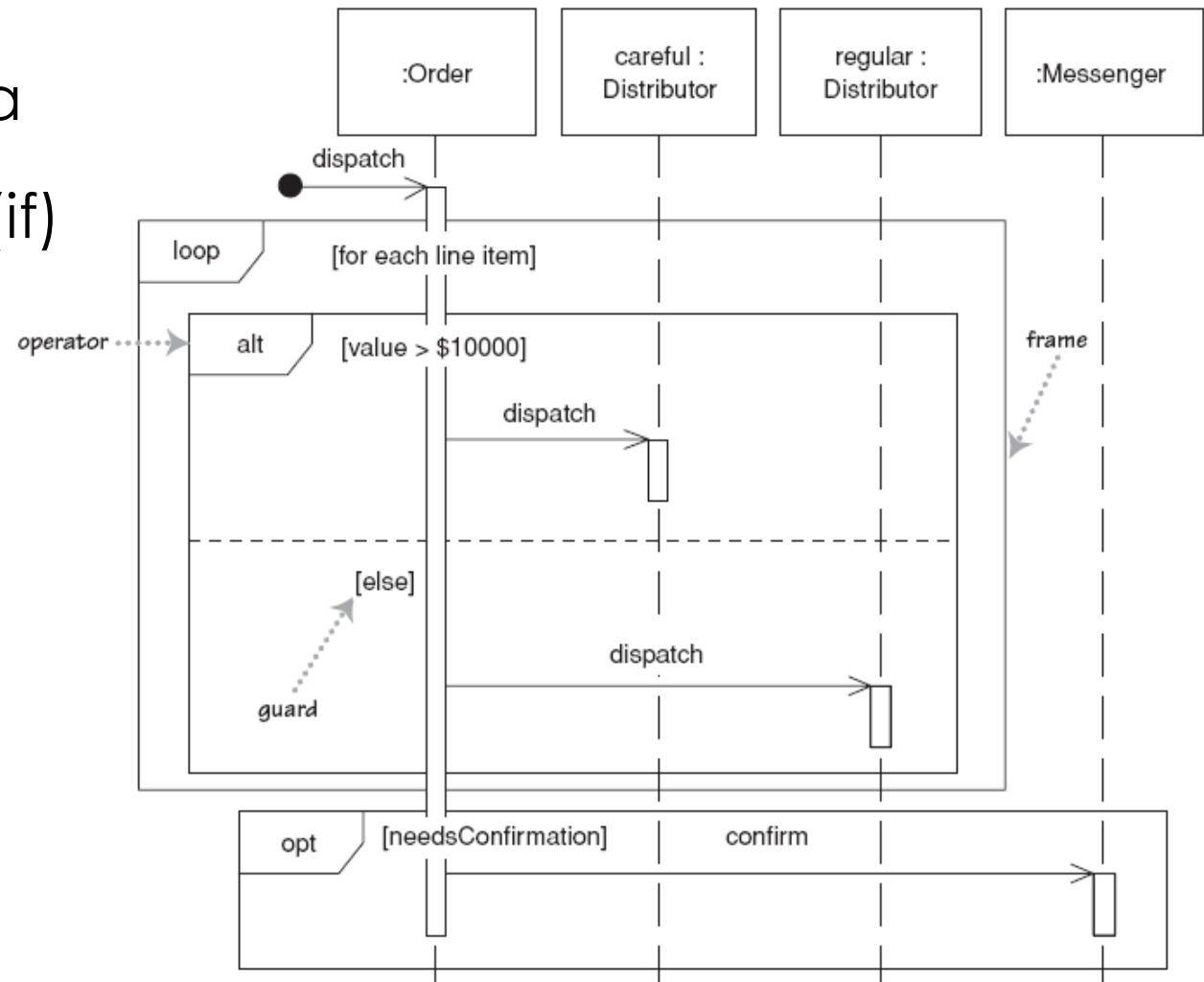
- **Aktivasi:**
  - Digambarkan sebagai kotak pada garis vertikal
  - Rekursif dapat digambarkan sebagai pemanggilan bersarang



# Seleksi, Pengulangan

- Diberikan bingkai: kotak pada diagram sekuens yang mengindikasikan ada seleksi (if) atau pengulangan (loop)

- `if` → (opt) [condition]
- `if/else` → (alt) [condition], dipisahkan garis putus-putus horisontal
- `loop` → (loop) [kondisi atau item yang diulang]

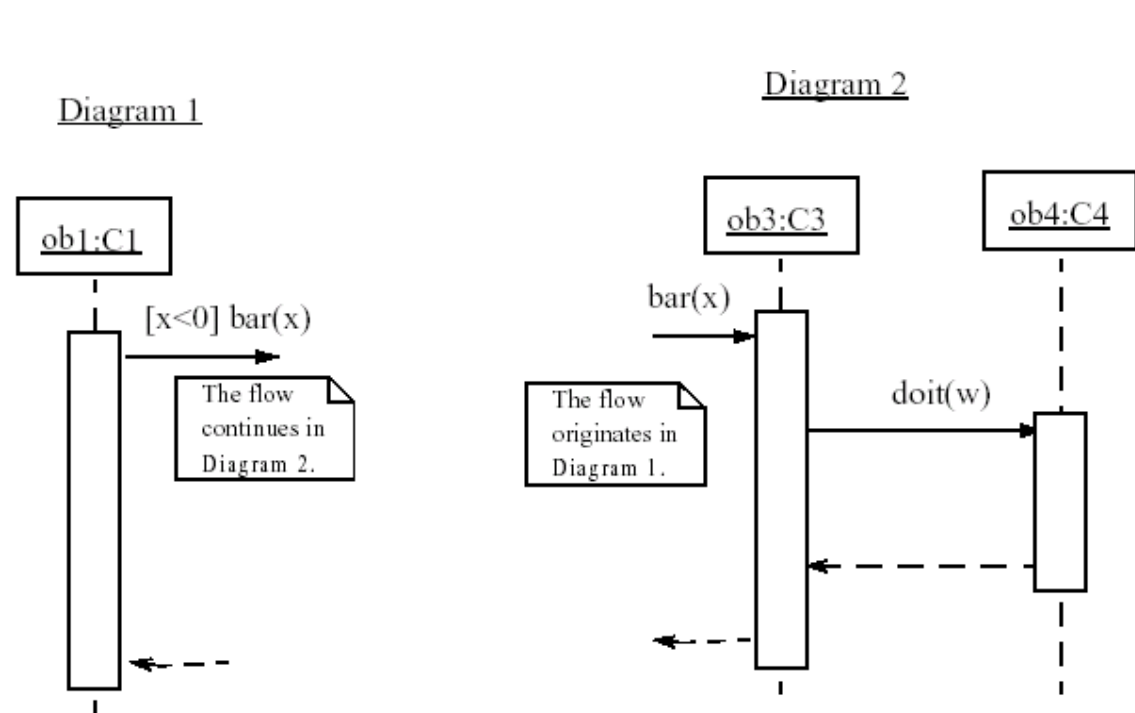
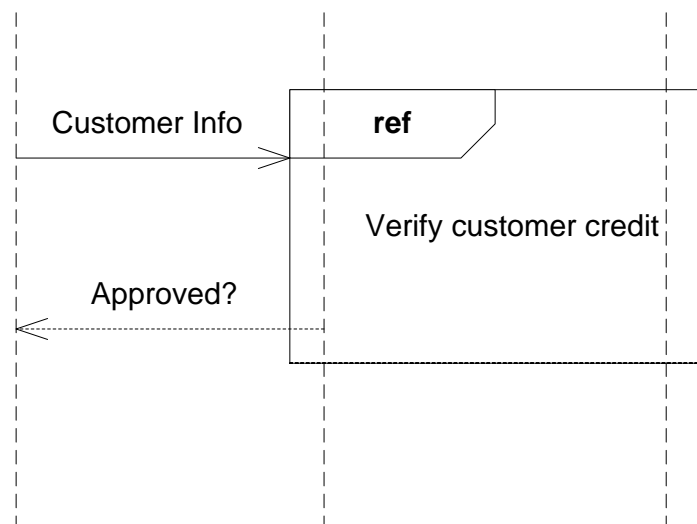


IF2250 - Sequence Diagram

# Menghubungkan Sekuens Diagram

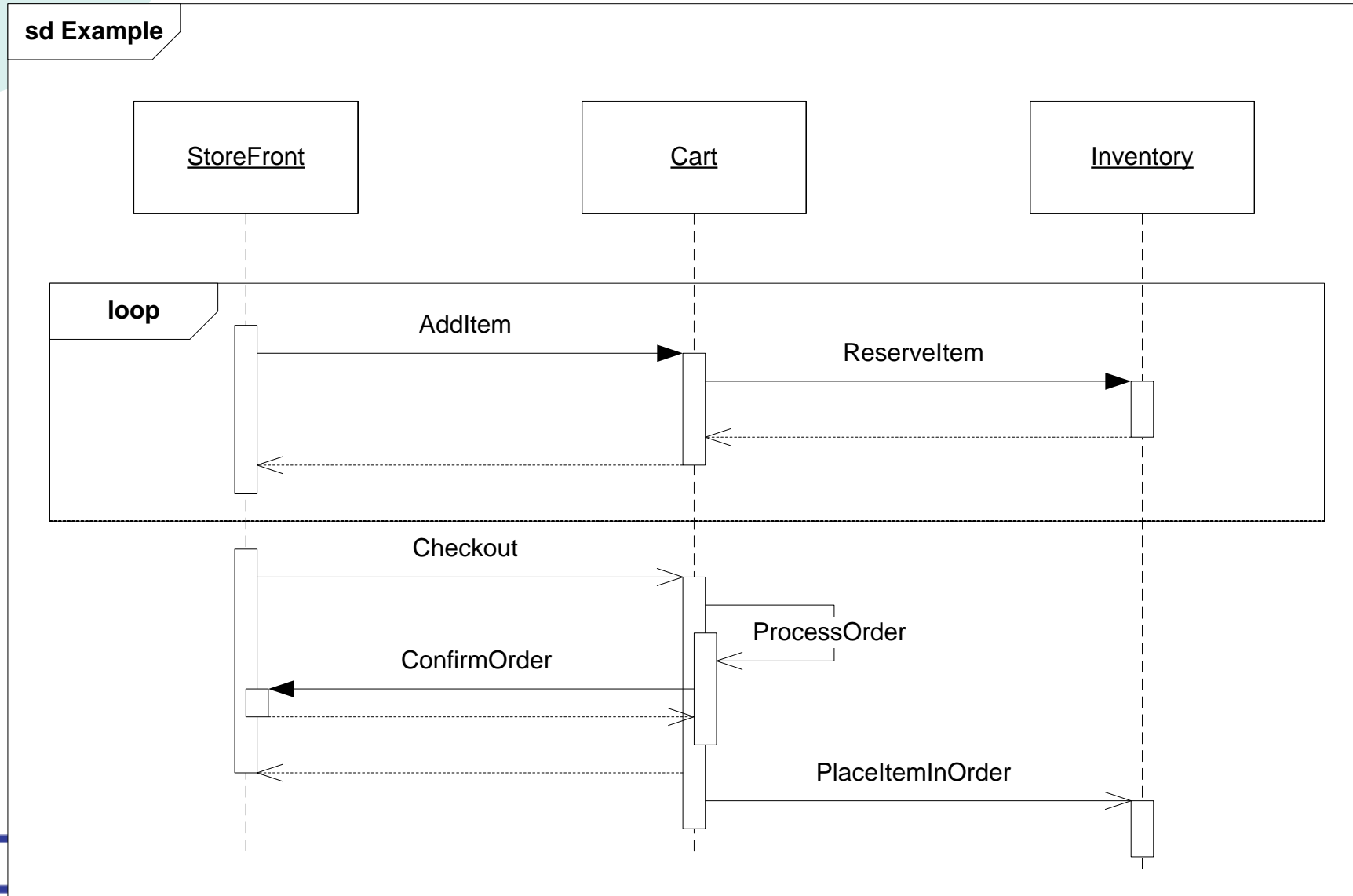
36

- Jika sekuens diagram terlalu besar, atau ingin mengacu ke sekuens diagram lagi, maka gunakan:
  - Panah dan juga komentar yang mengacu ke diagram lain
  - Buat bingkai “**ref**” yang diikuti dengan isi nama diagram lain



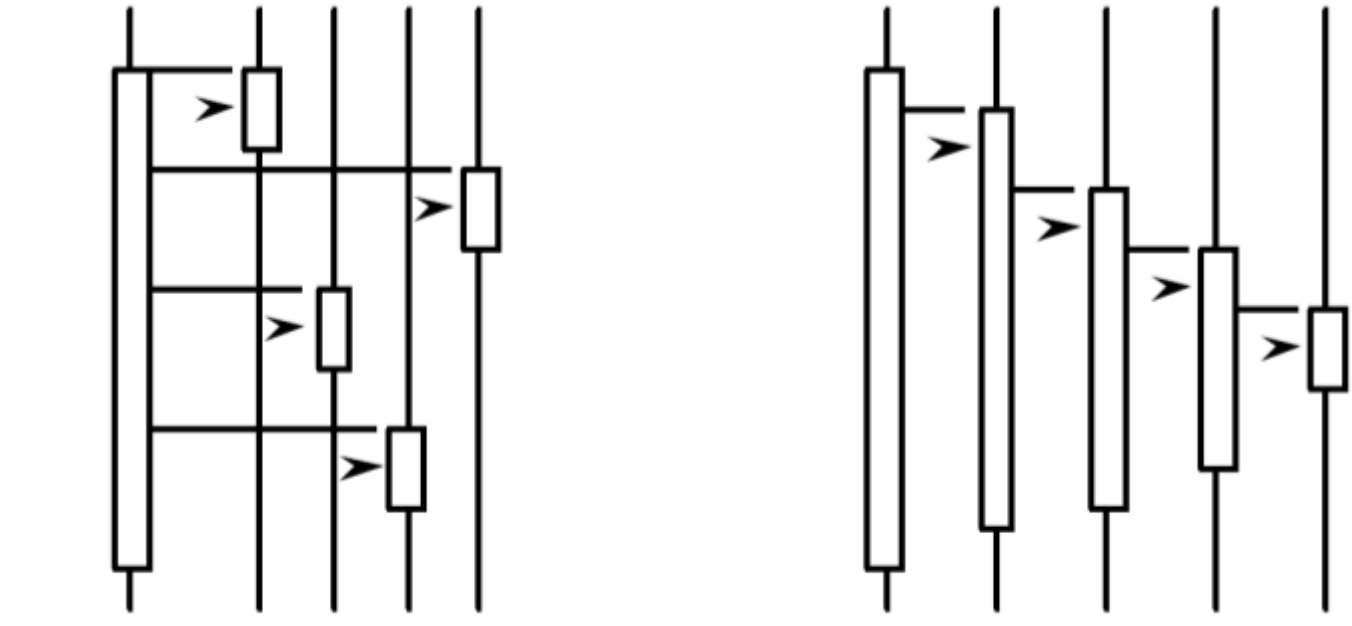
# Contoh diagram sekuens

37



# ***Bentuk-bentuk Diagram Sekuens***

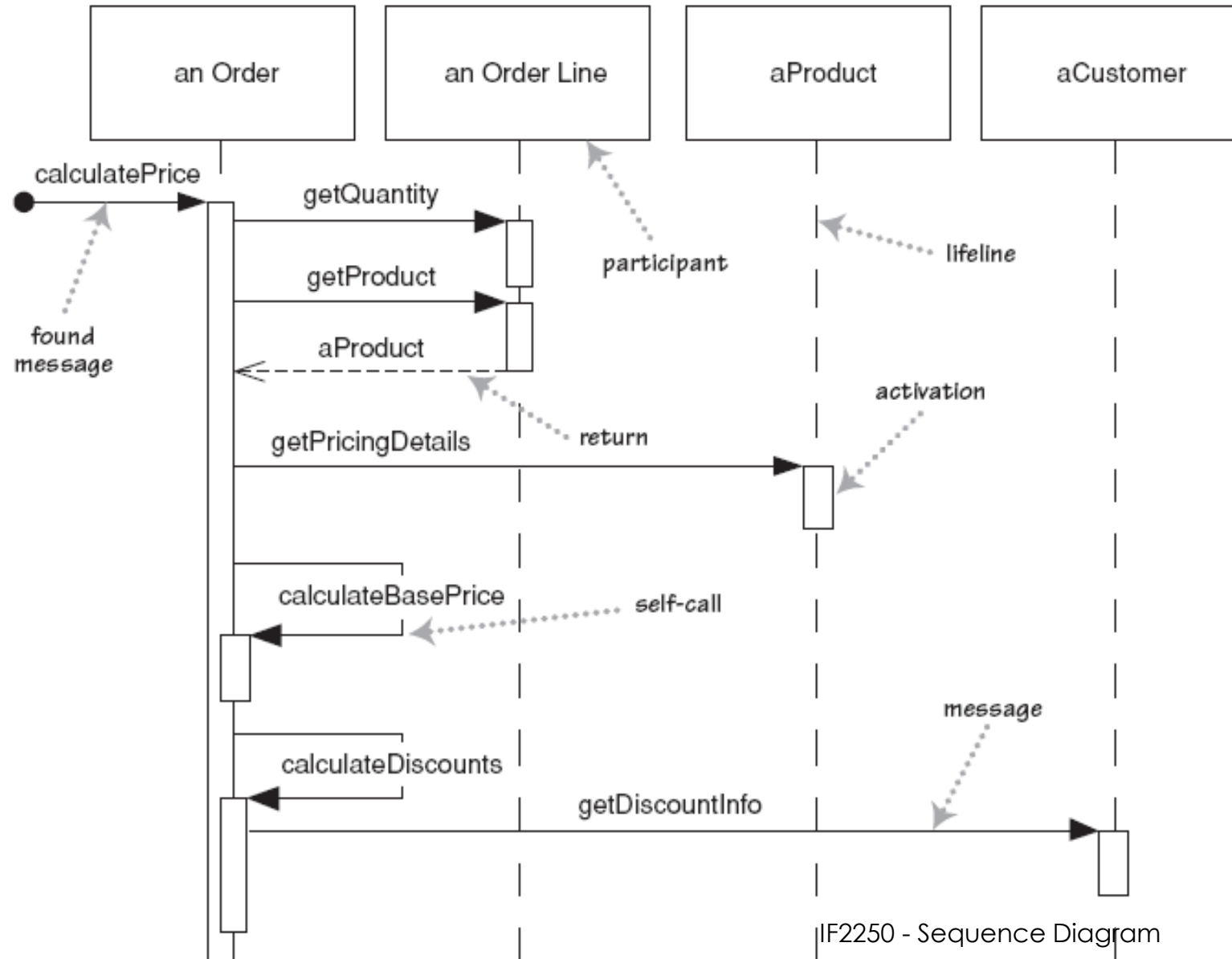
- Lihat aliran panah pada gambar berikut!
  - Terpusat?
  - Tersebar?



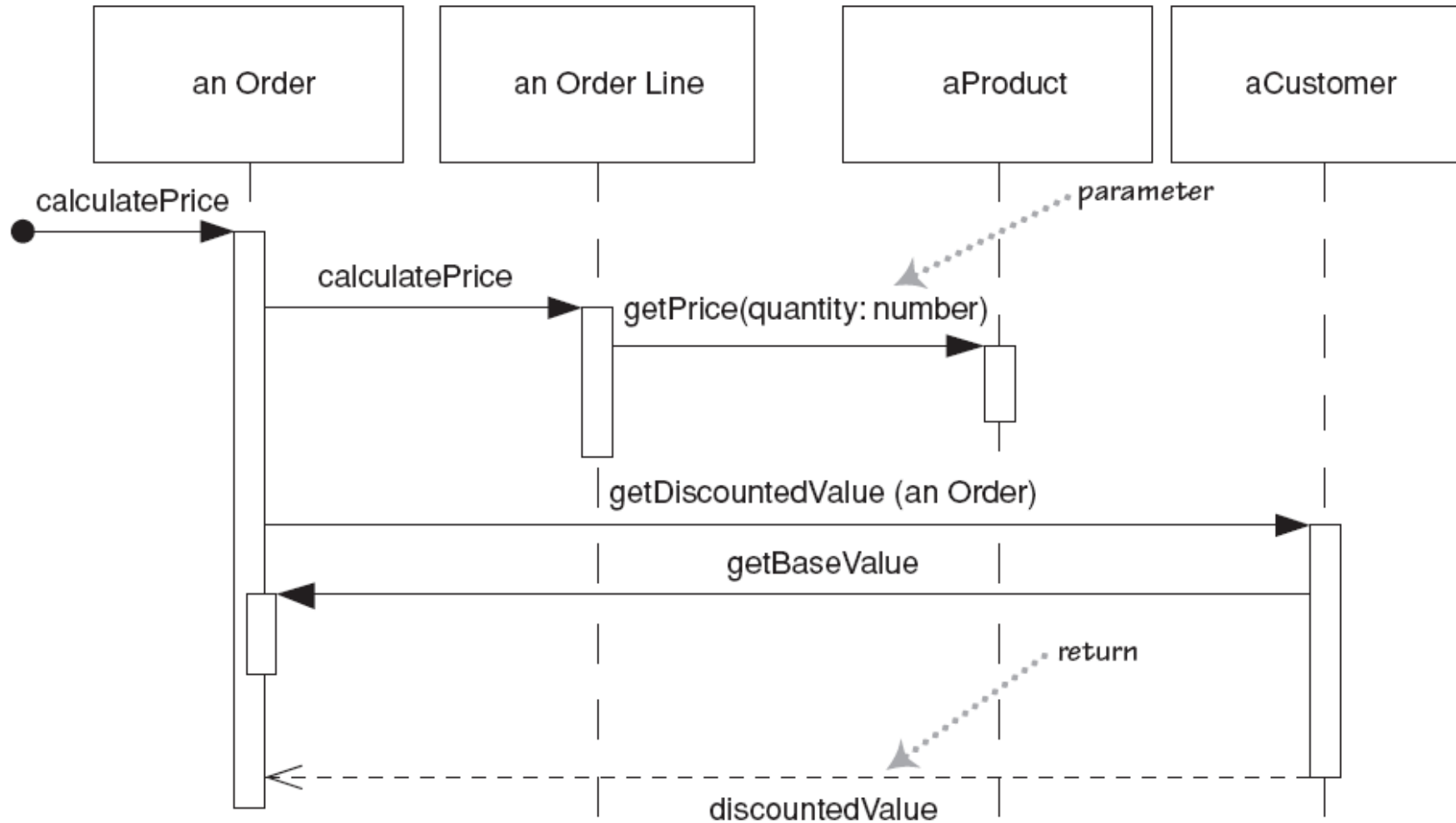
- Pada sistem terpusat (centralized), 1 peserta melakukan seluruh pemrosan
- Pada sistem tersebar (distributed), pemrosesan dibagi ke peserta lain

# Contoh Sekuens Diagram

39



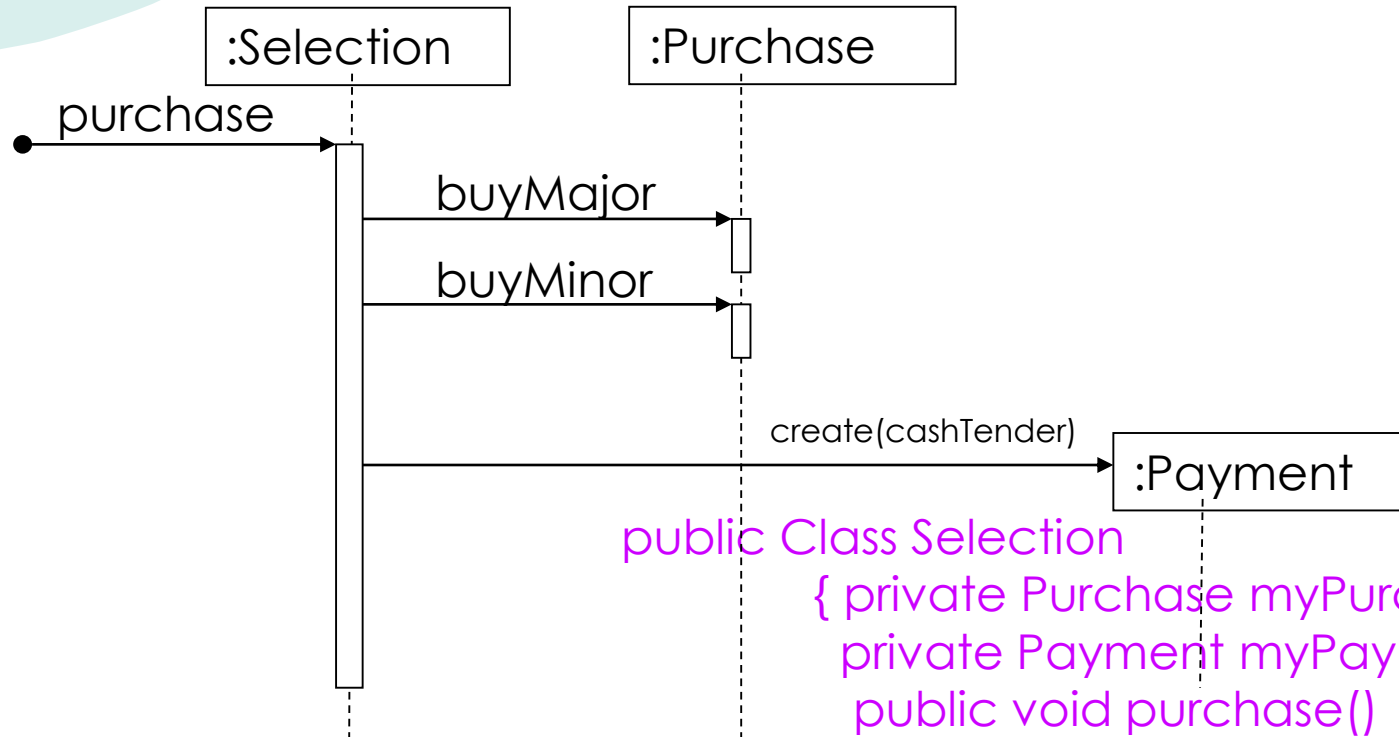
# Contoh Sekuens Diagram





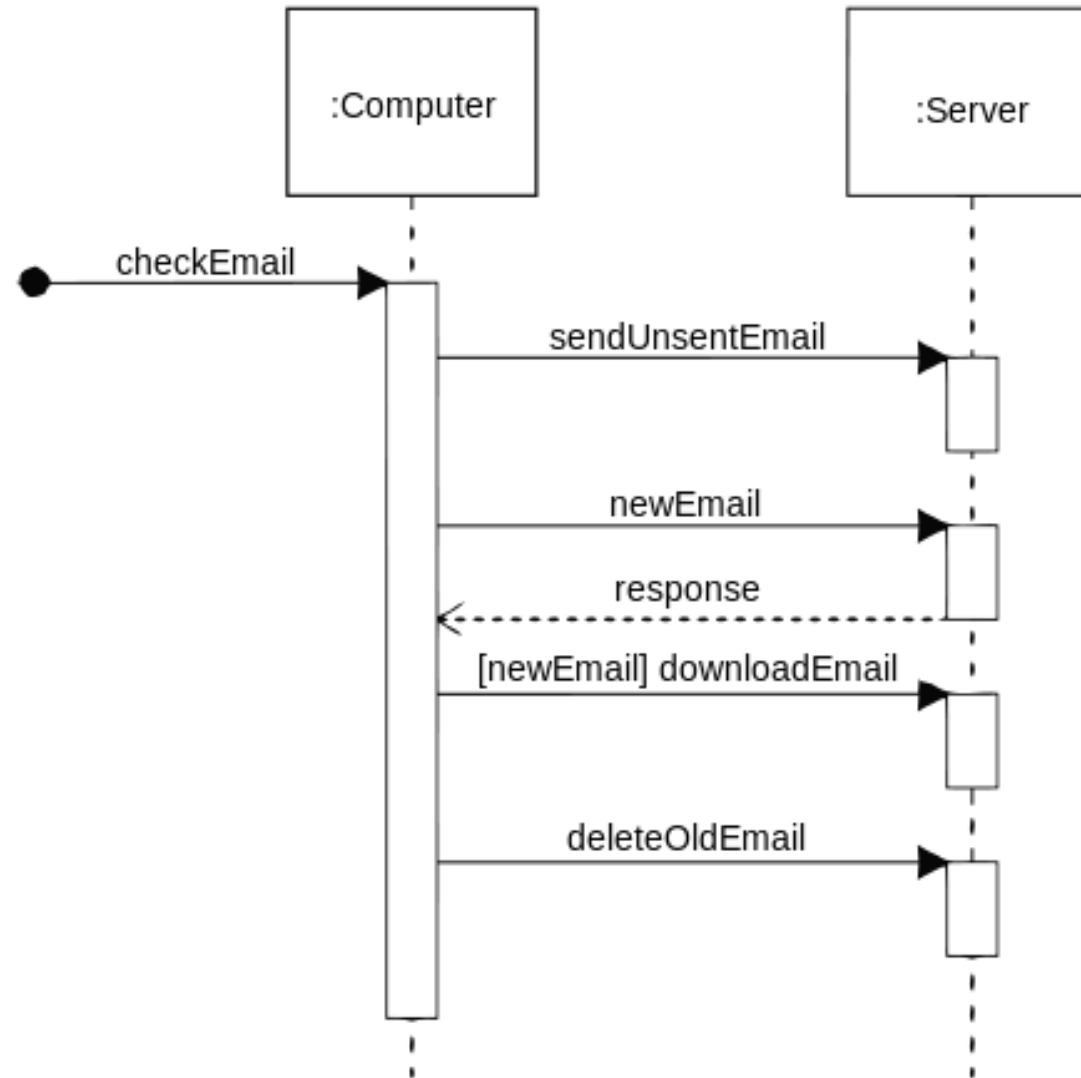
# *Dari Sekuens Diagram ke Pemrograman*

41

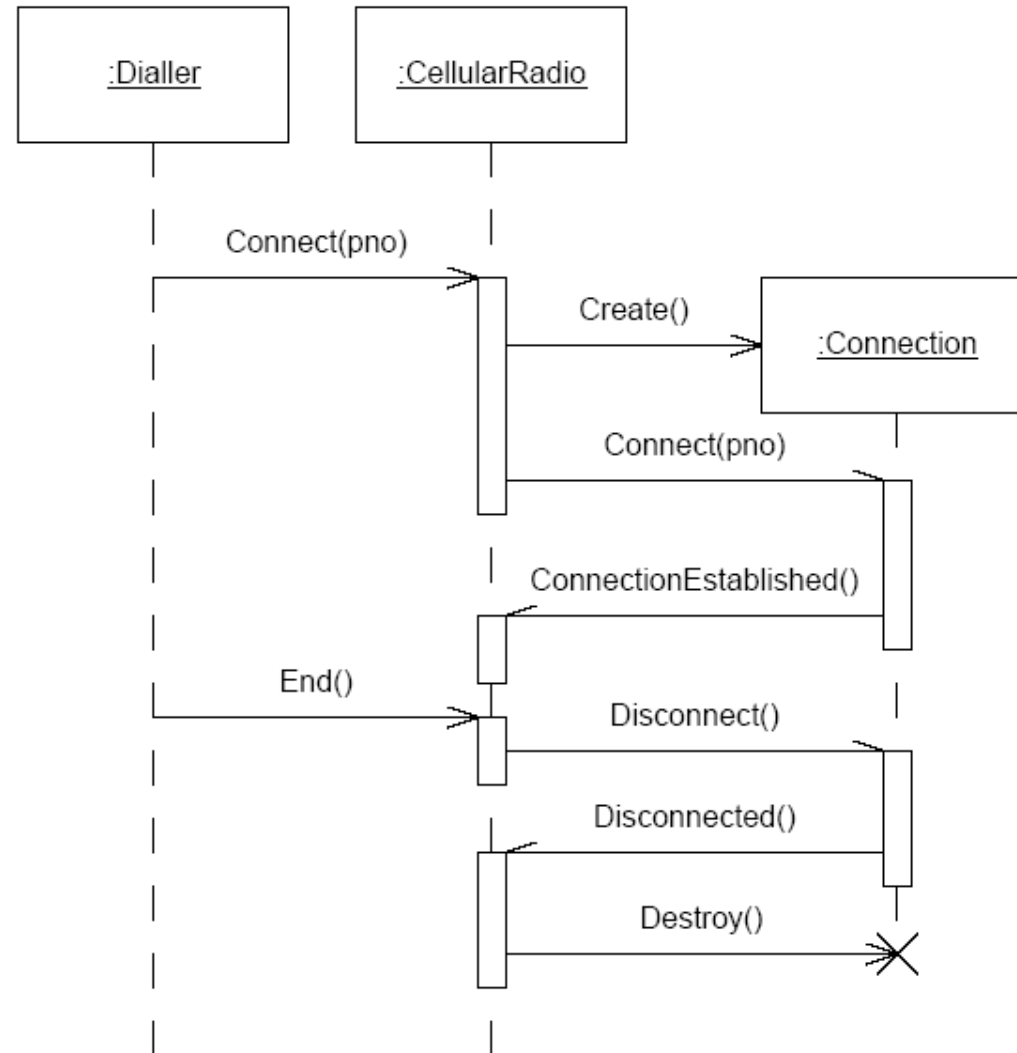


```
public Class Selection
{ private Purchase myPurchase = new Purchase();
  private Payment myPayment;
  public void purchase()
  { myPurchase.buyMajor();
    myPurchase.buyMinor();
    myPayment = new Payment( cashTender );
    //..
  }
  // ..
}
```

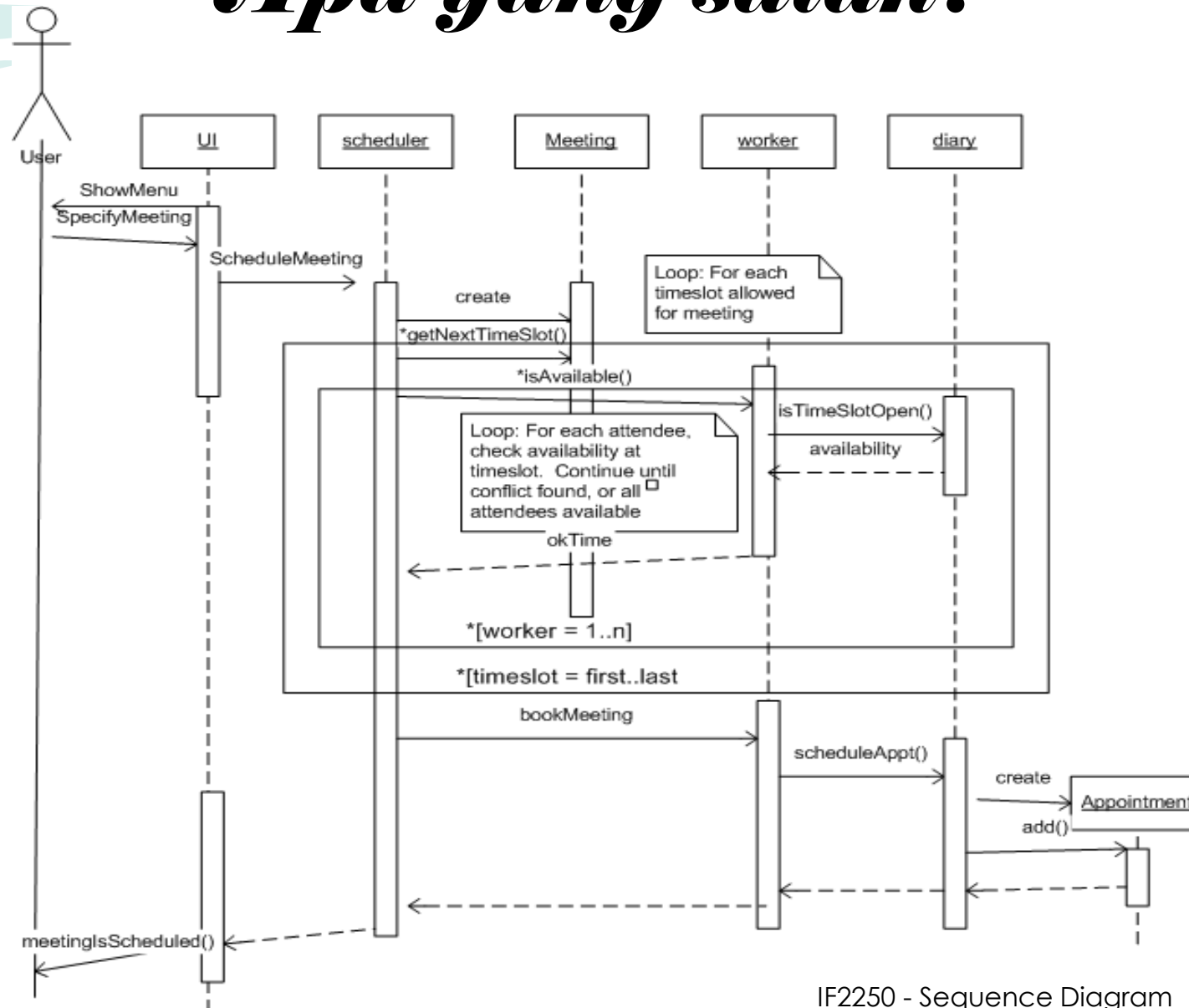
# *Bagaimana Cara Membaca Diagram berikut?*



# *Apa yang salah?*



# *Apa yang salah?*



IF2250 - Sequence Diagram

# ***Kenapa tidak dibuat algoritma saja?***

- Sekuens Diagram lebih mudah dimengerti untuk pengguna
  - Tidak semua bagian kode program ditampilkan, sehingga mudah dimengerti
  - Mudah digunakan untuk diskusi dalam 1 tim ataupun dengan pihak lain
- Keterlibatan Objek/Kelas dapat terlihat pada halaman yang sama
- Tidak tergantung Bahasa



# *Contoh Diagram Sekuens*

## *[Fowler]*

Ada pesanan datang, sistem akan memanggil suatu perintah untuk menghitung harganya. Pesanan ini memiliki daftar pesan. Lalu untuk menentukan harganya, akan diambil dari data produk. Kemudian dari pesanan tersebut, akan dihitung total harganya, sesudahnya potongan harga akan diberikan berdasarkan kategori pelanggan



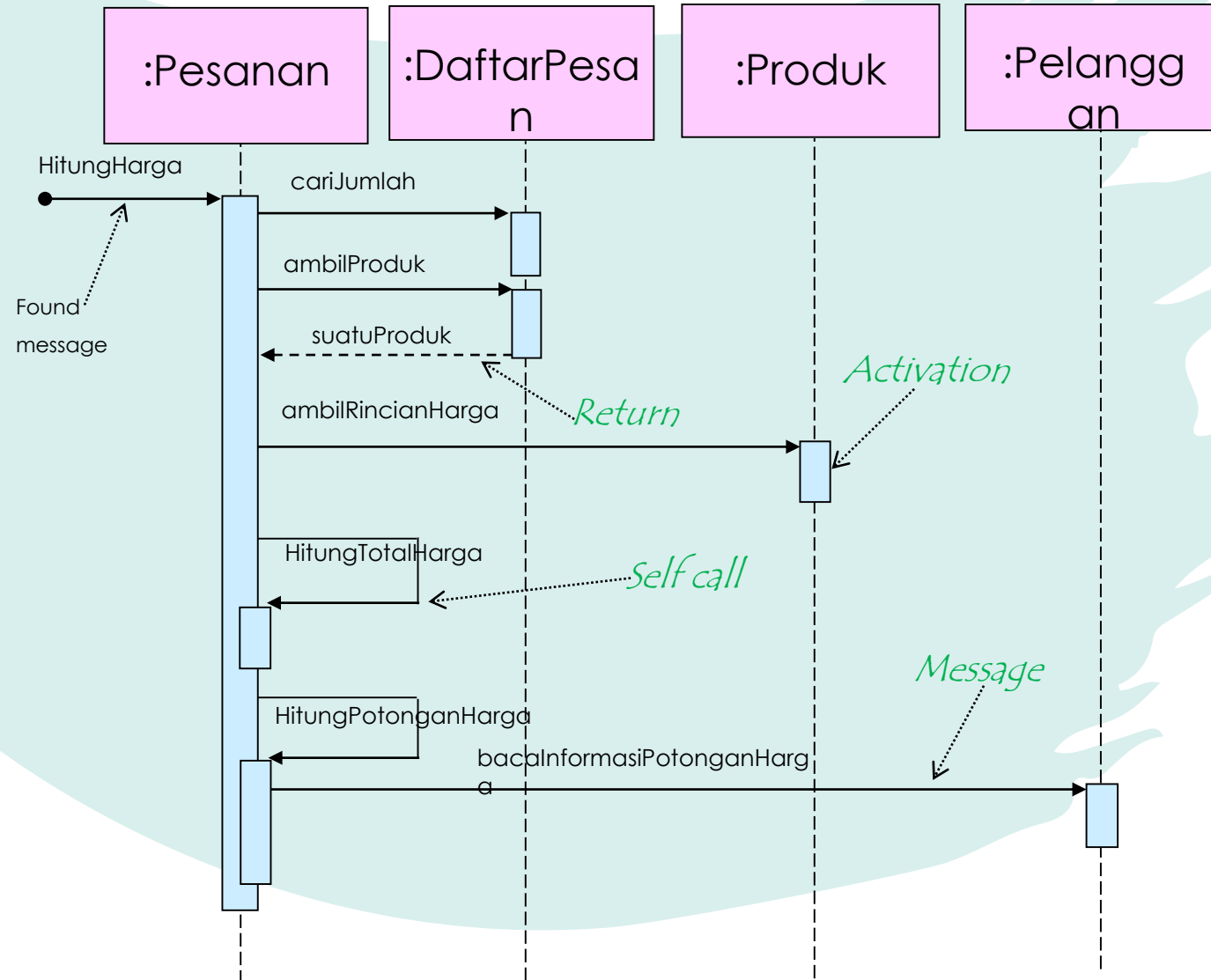
# *Contoh Diagram Sekuens*

## *[Fowler]*

Ada **pesanan** datang, sistem akan memanggil suatu perintah untuk menghitung harganya. Pesanan ini memiliki **daftar pesan**. Lalu untuk menentukan harganya, akan diambil dari data **produk**. Kemudian dari pesanan tersebut, akan dihitung total harganya, sesudahnya potongan harga akan diberikan berdasarkan kategori **pelanggan**



# Contoh Diagram Sekuens





# *Contoh Pembuatan Diagram Sequence*



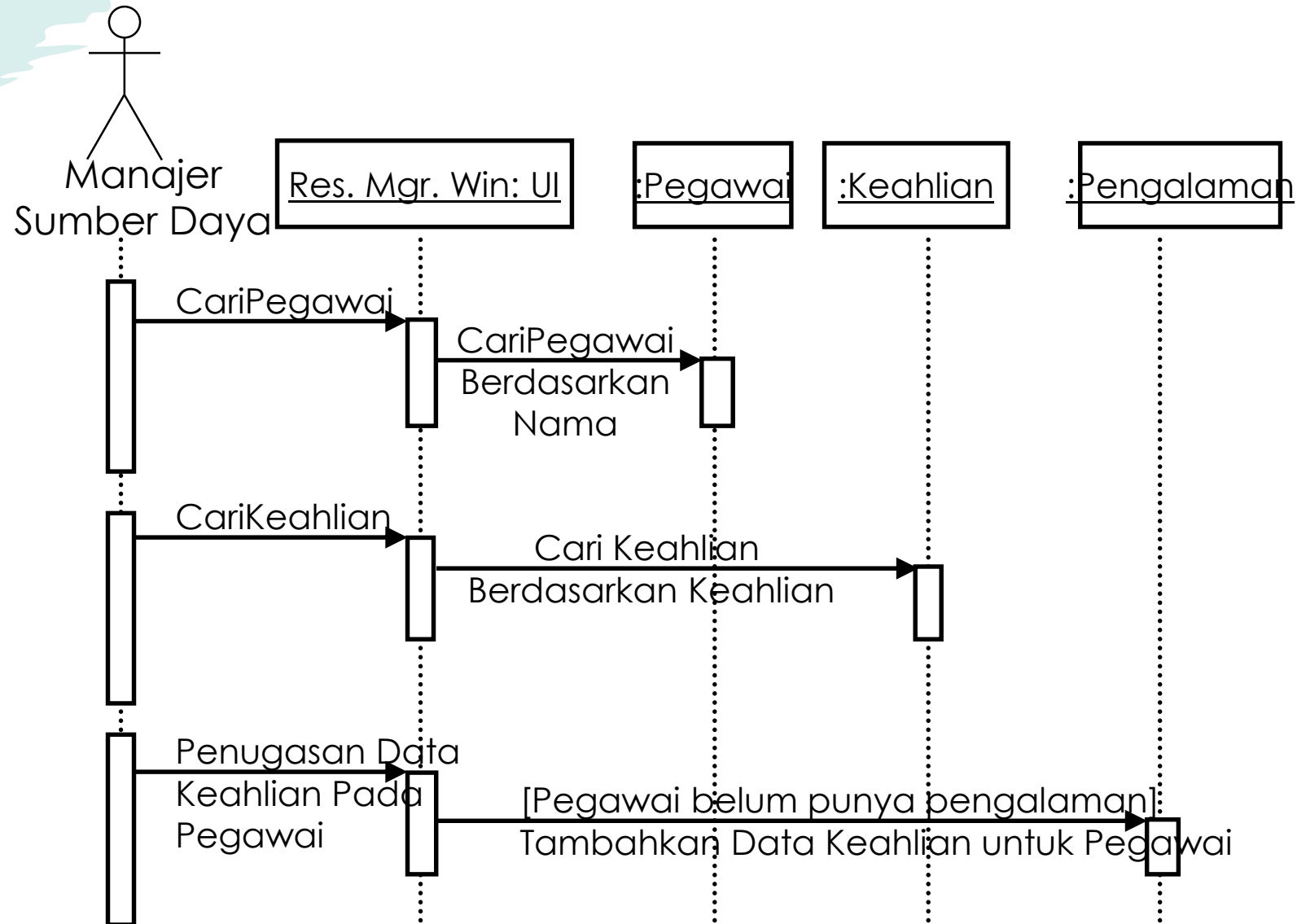
# *Urutan Langkah*

- **Gambarkan Diagram Sekuens (Sequence Diagram)**
  - Satu 'Use-Case' memiliki satu diagram sekeuns
  - Pada diagram sekuens akan memperlihatkan tugas suatu kelas
- Tambahkan metode/operasi dari diagram sekuens ke setiap kelas terkait



# Sekuens Diagram – Memasukkan Data Keahlian Pegawai

51



# *Urutan Langkah*

- Analisa kebutuhan dari Pengguna
  - Amati 'Kata Benda' -> kandidat kelas
  - Amati 'Kata kerja' yang terkait kata benda → kandidat asosiasi antar kelas, atau kandidat operasi dalam kelas
  - Gambar Diagram Kelas
    - Tentukan atribut
  - Gambarkan Diagram objek untuk membantuk menjelaskan diagram Kelas
- Cari 'Use Case'
  - Identifikasi Aktor
  - Identifikasi Use Case
  - Cari hubungan antara Aktor, juga antara use-case
  - Gambarkan diagram Use-Case
- Gambarkan Diagram Sekuens (Sequence Diagram)
  - Satu 'Use-Case' memiliki satu diagram sekeuns
  - Pada diagram sekuens akan memperlihatkan tugas suatu kelas
- **Tambahkan Metode/Operasi pada Diagram Kelas**



# ***Tambahkan Metode/Operasi***

- Baca lagi Diagram Sekuens untuk mencari atau menambahkan metode/operasi pada suatu kelas

Pegawai
Nama
+ static Pegawai cariPegawai (nama: string); + static list of Workers getPegawai();

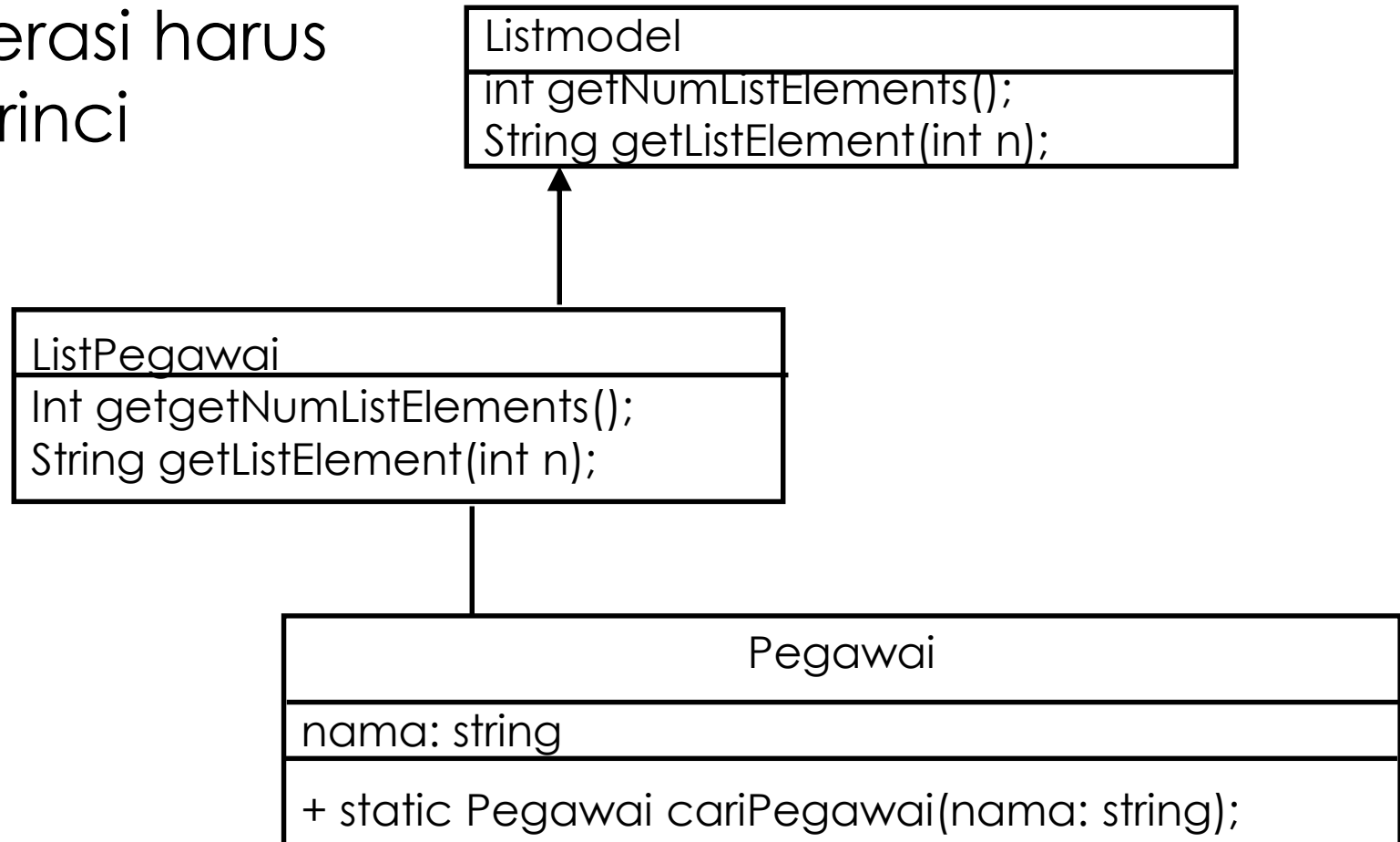
# ***Tahapan Perancangan***

- Metode/Operasi harus dibuat lebih rinci

<b>Pegawai</b>
Nama : string
+ static Pegawai cariPegawai (nama: string); + static int getJumlahPegawaiNWorkers(); + static Pegawai getPegawai(int);

# *Tahapan Perancangan*

- Metode/Operasi harus dibuat lebih rinci

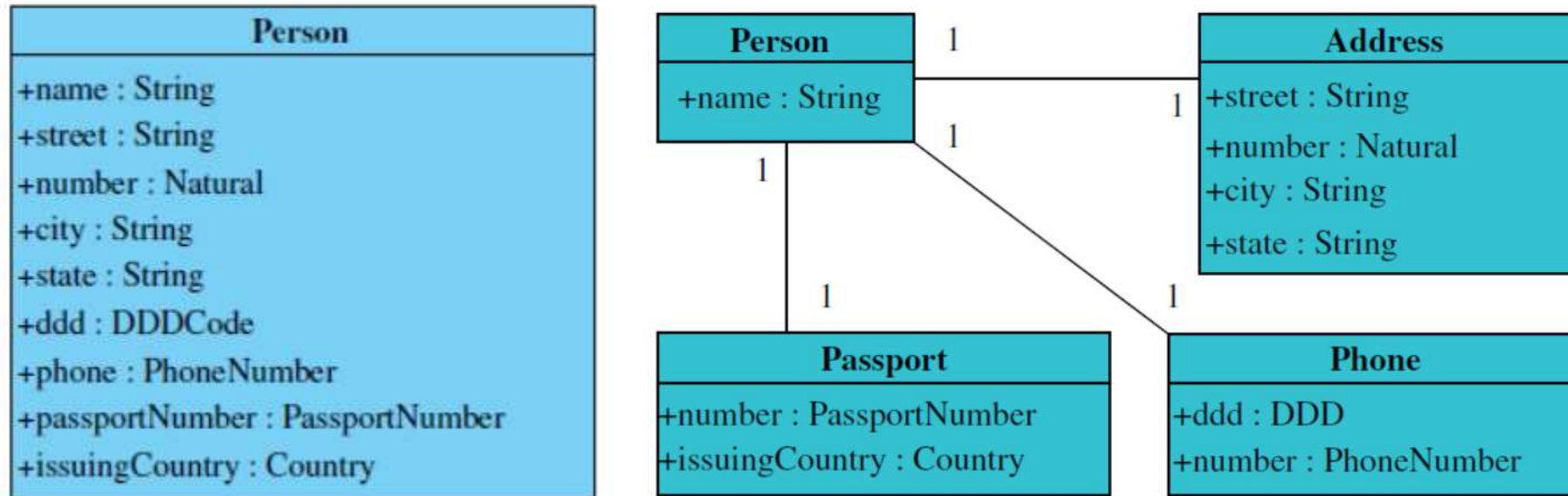


# ***Masalah- masalah pada Rancangan Kelas (Class Design)***

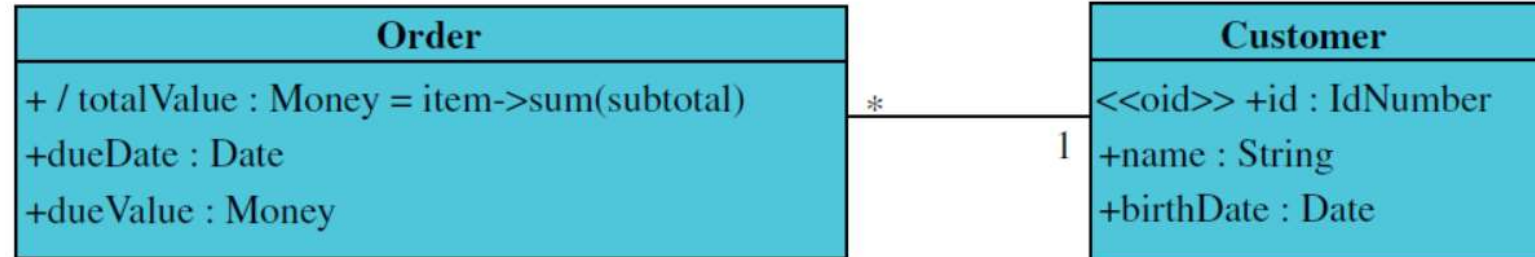
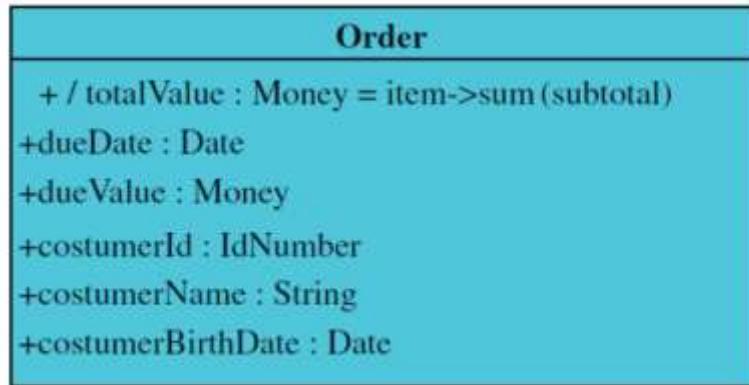




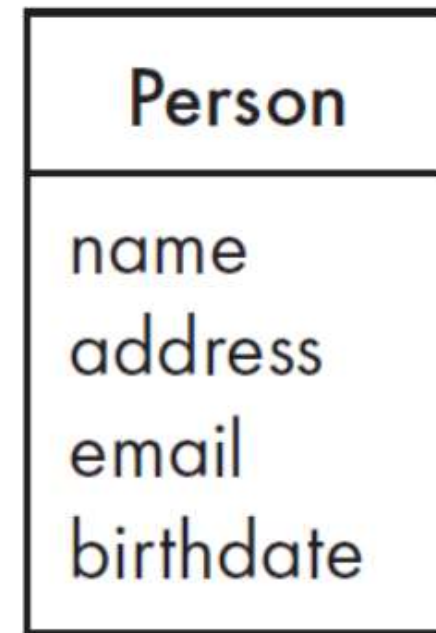
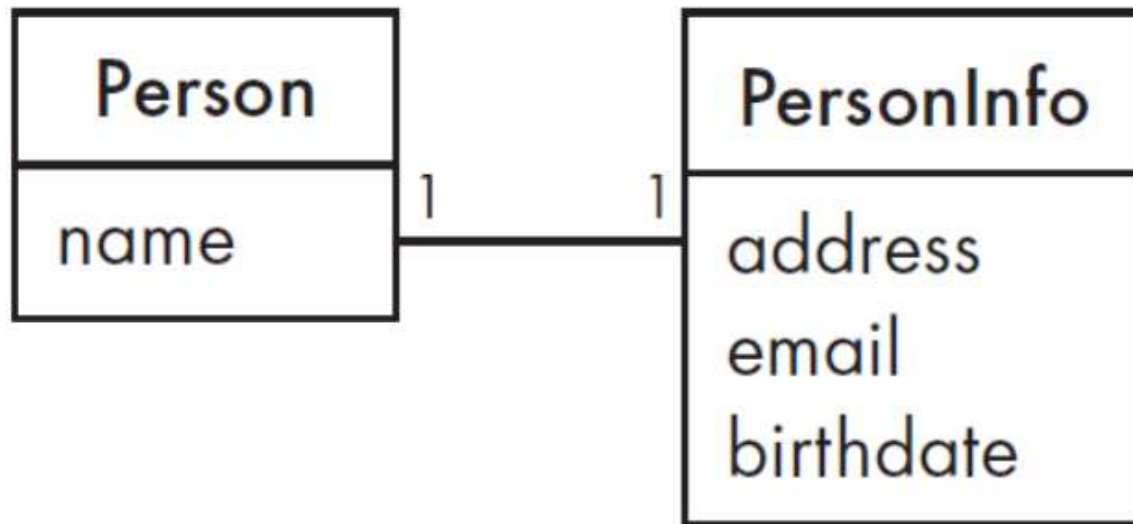
# *Apa masalah yang kiri dengan yang kanan?*



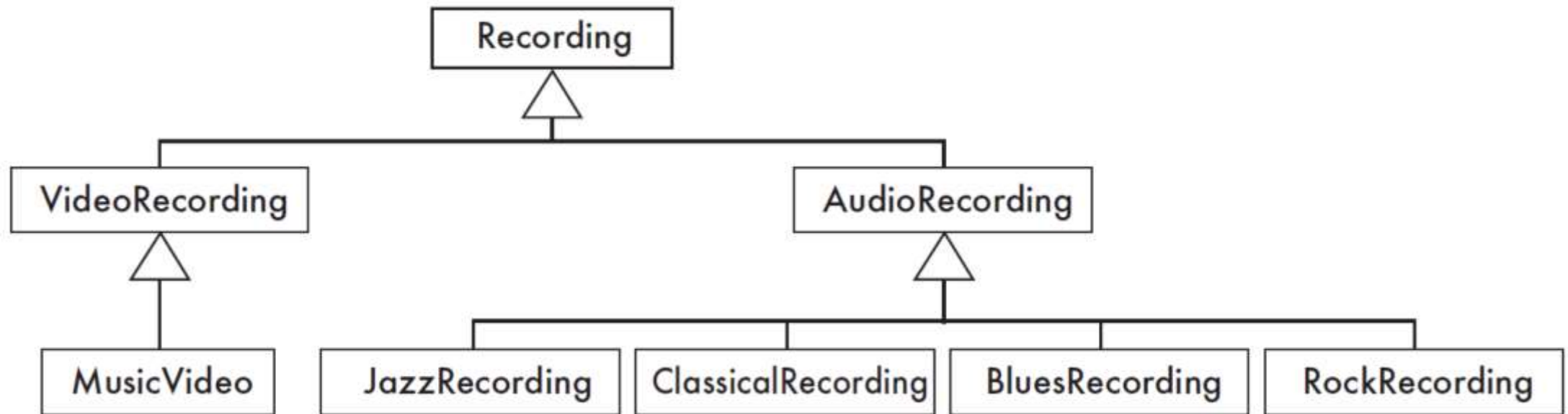
***Apa masalah yang kiri atas dengan yang kanan bawah?***



# ***Kenapa yang kanan lebih baik dari yang kiri?***

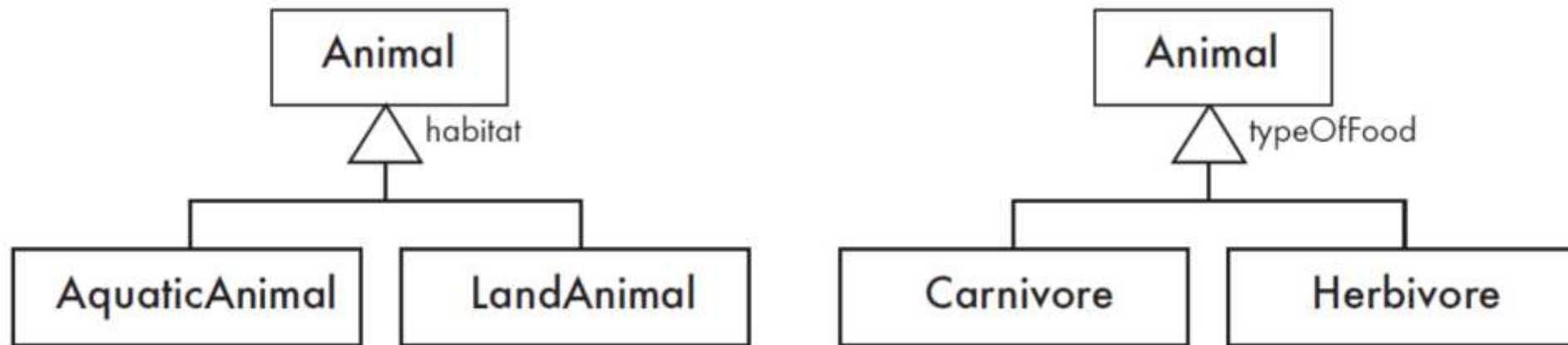


# ***‘Audio Recording’ mungkin tidak perlu diturunkan. Kenapa?***



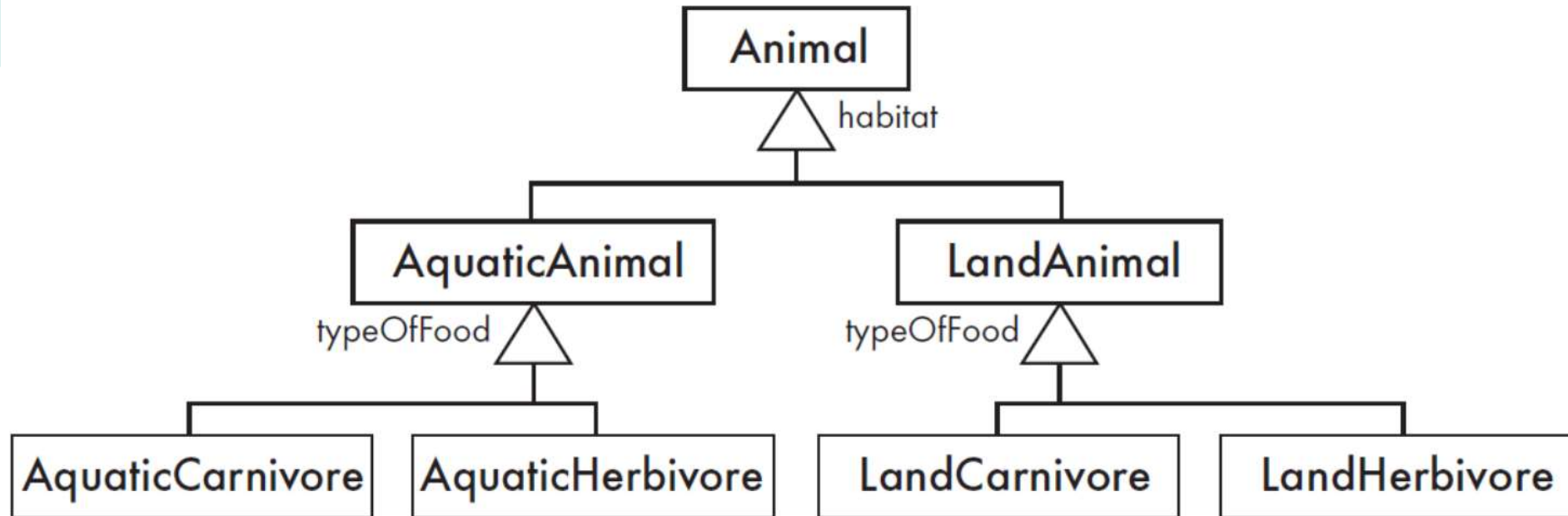
# ***Masalah Dua Generalisasi***

- Sebagai contoh, Pada kelas 'Animal' mungkin dapat dibagi berdasarkan habitat nya (darat atau air), dan Tipe makanannya (karnivora atau herbivora)



Masalahnya, jika ini diimplementasikan maka akan memiliki jumlah objek yang banyak untuk satu instansiasi hewan

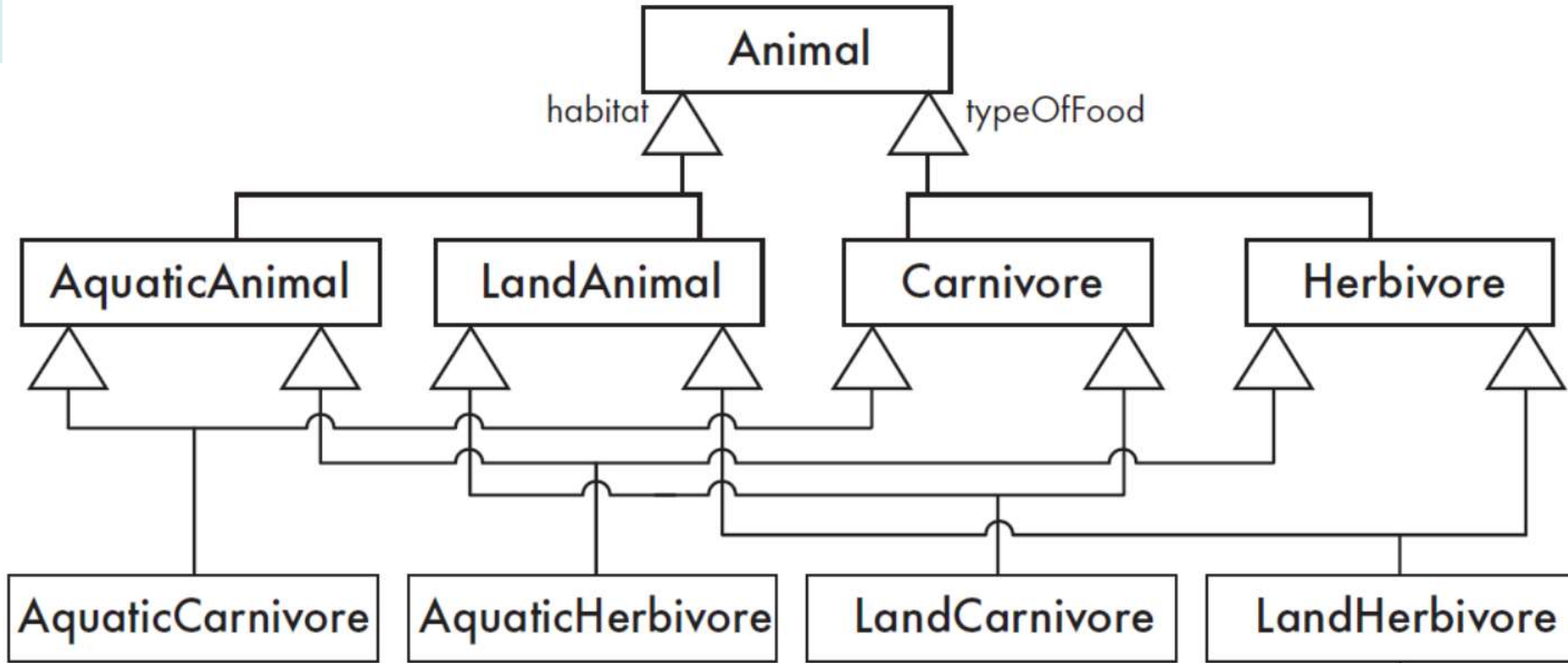
# *Bagaimana jika disatukan?*



- Dengan solusi ini, maka karakteristik/atribut dari generalisasi yang kedua (*landAnimal*), akan harus diduplikasi
- Jumlah kelas juga akan lebih banyak – menambah kompleksitas

# *Bagaimana jika dibuat Multiple Inheritance*

63

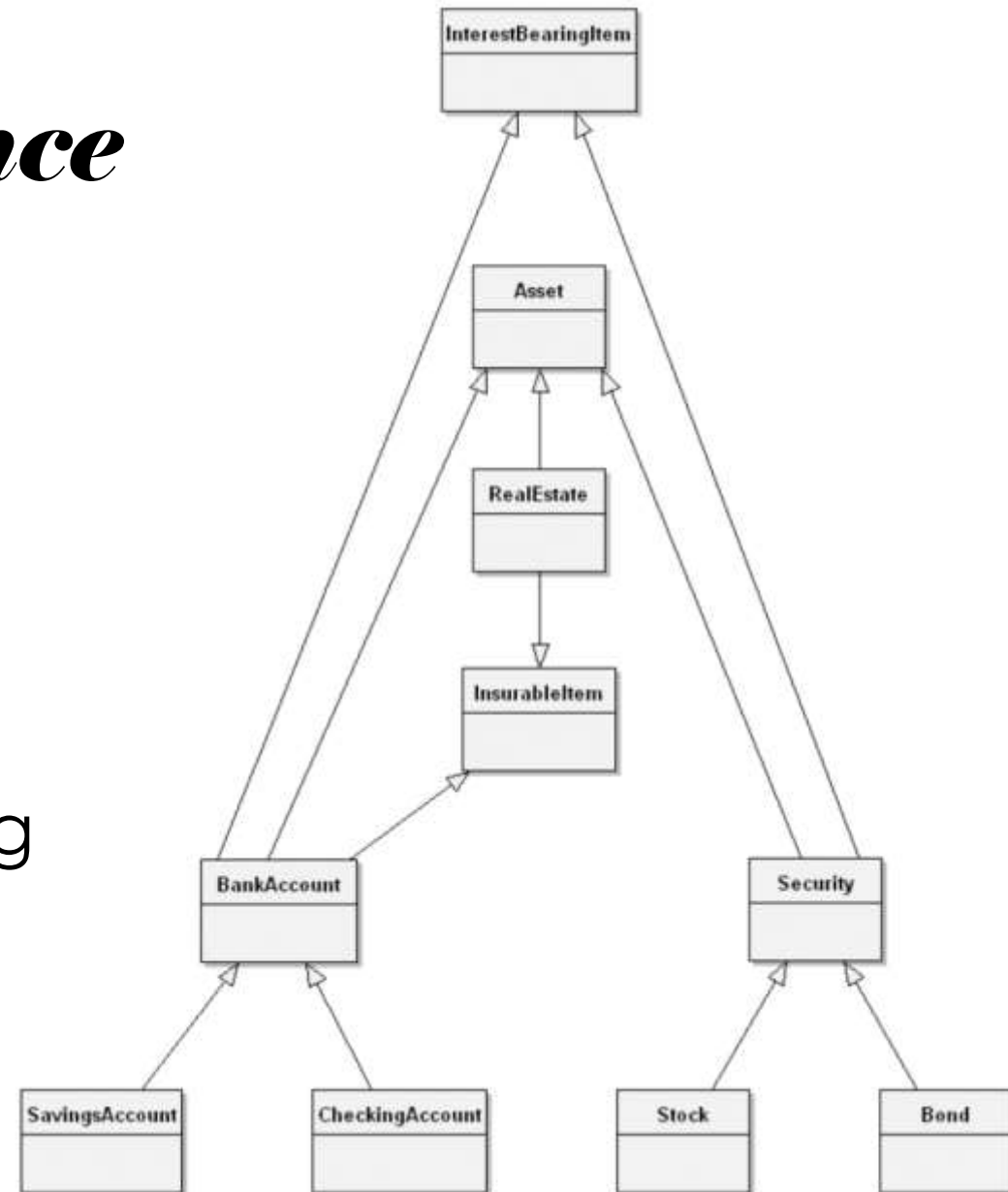


- Kombinasi ini akan menjadi kompleks, sehingga patut dihindari
- Java tidak mengenal multiple inheritance

# *Multiple Inheritance*

Apa saja masalahnya?

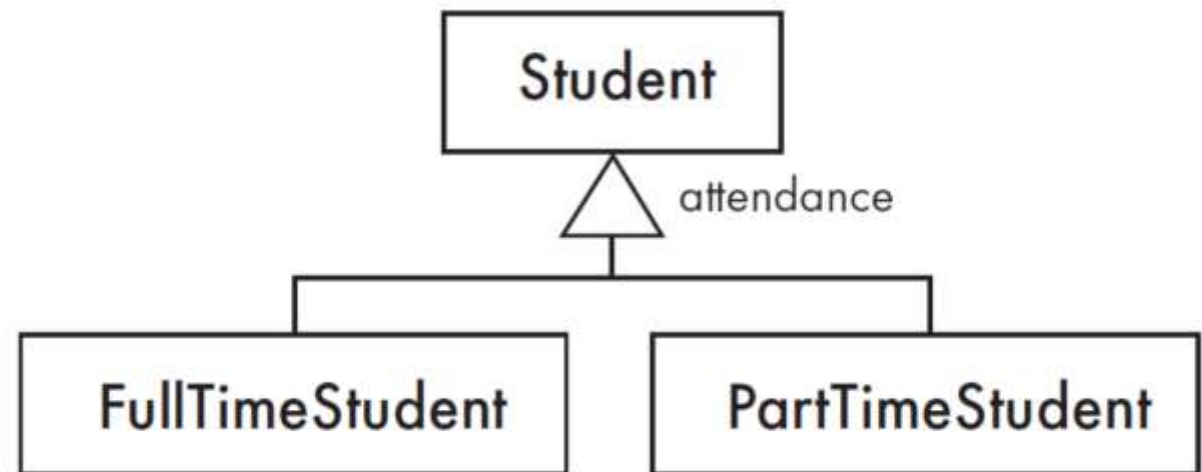
- Implementasi sulit
- Kompleks!
- Bahasa Pemrograman mungkin tidak mendukung
  - Contoh Java





# Masalah Objek yang berubah menjadi Objek lain

- Suatu sekolah dimungkinkan seorang mahasiswa berpindah status dari Full-time menjadi Part-Time.
- Kelas desainer kemudian membuat diagram kelas sebagai berikut.
- Apa masalahnya?  
Bagaimana solusinya?

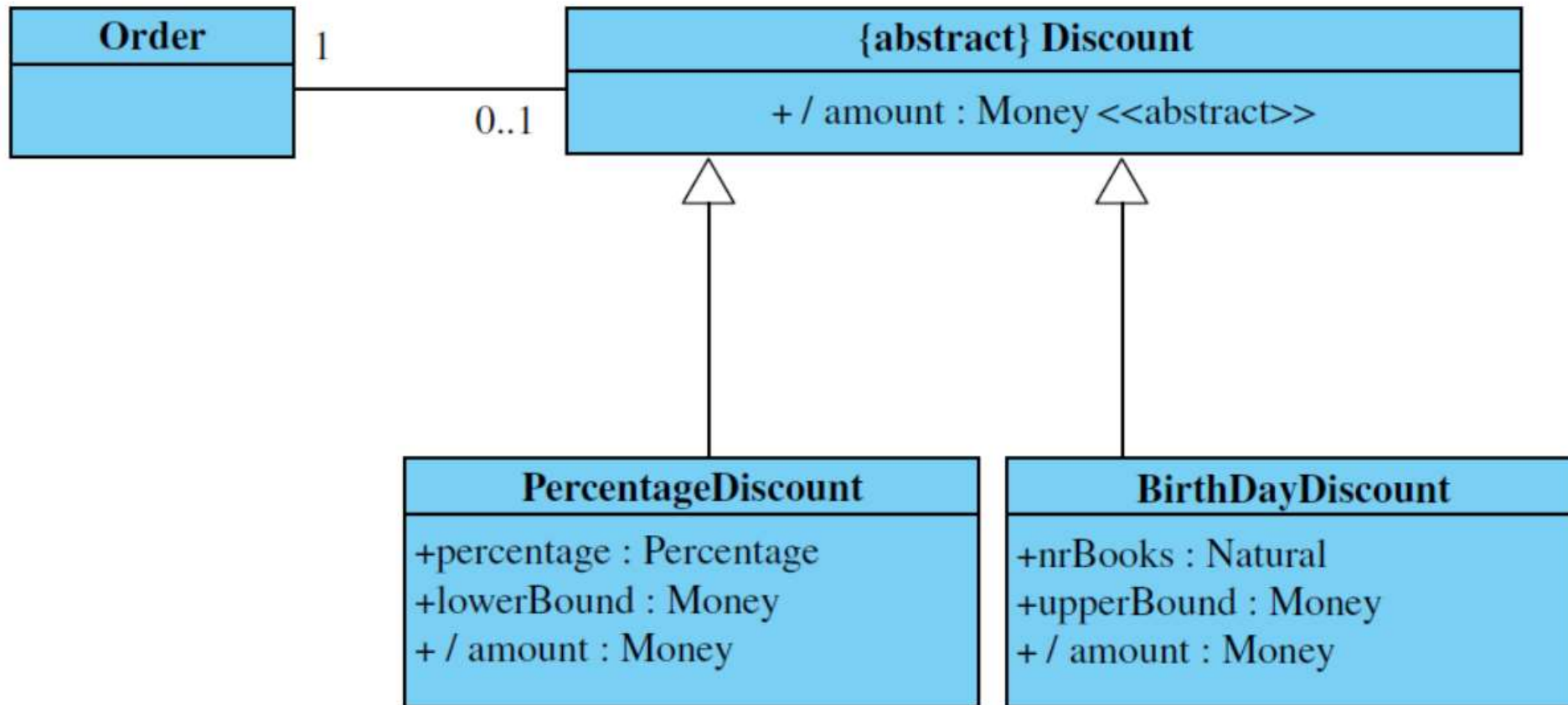


Buat menjadi atribut tambahan pada Student!

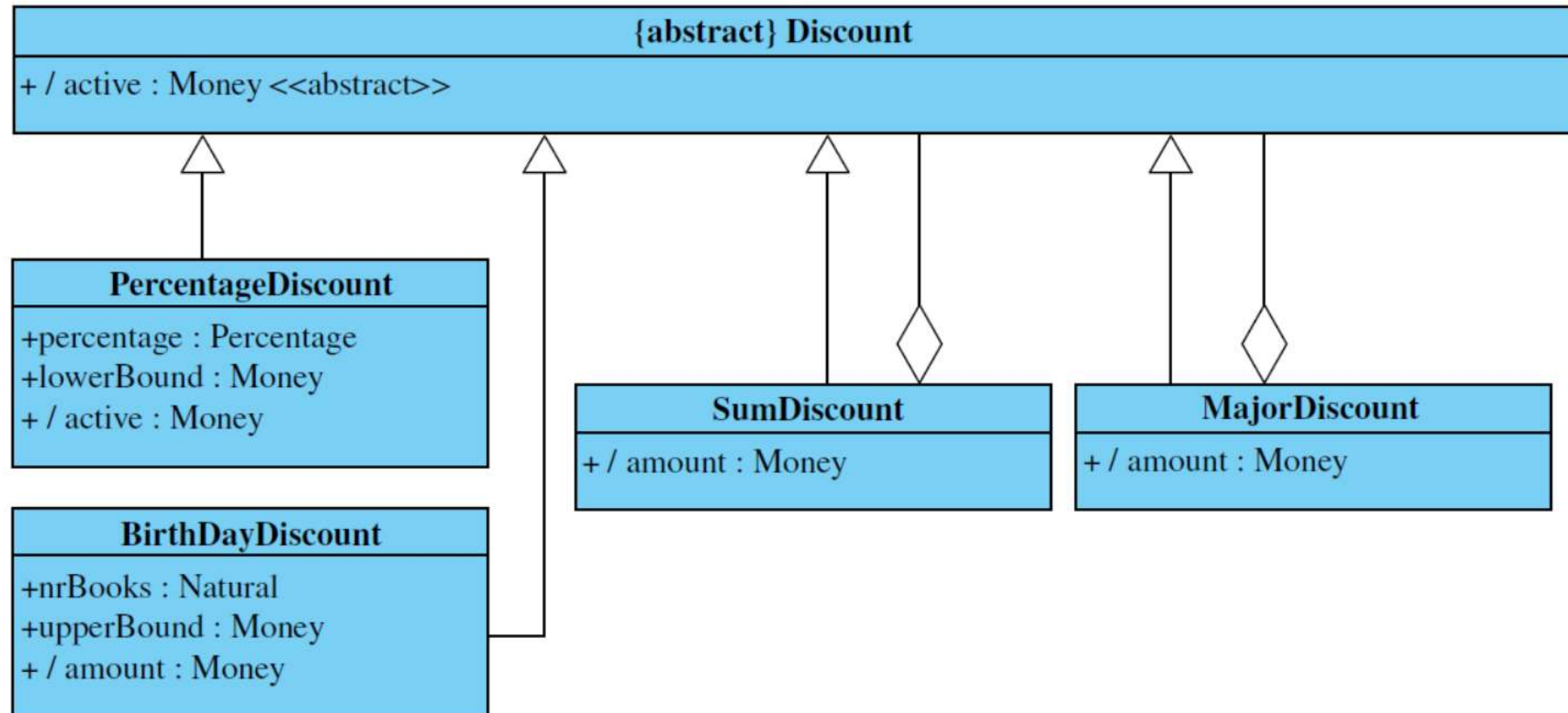
# *Bagaimana merepresentasikan Kelas yang mungkin memiliki variasi?*

## *Gunakan Kelas Abstract*

66



# Variasi lain: Composite Pattern



- Baca:

- Chapter 1-3, Grady Booch et.al, “Object Oriented Analysis and Design 3<sup>rd</sup> Edition”, Addison Wesley, 2007
- Gomaa, Hassan. Software modeling and design: UML, use cases, patterns, and software architectures. Cambridge University Press, 2011.
- Fowler, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.

