Use Case Diagram

Mata Kuliah Testing & Implementasi Sistem
Program Studi Sistem Informasi 2013/2014
STMIK Dumai
-- Pertemuan 5 --

Acknowledgement

Main materials:

- [Pressman, 2010] Pressman, Roger S. Software Engineering: A Practitioner's Approach. New York:McGraw-Hill Higher Education, 2010. Print

Supplements:

- [Yud, 2012] Yudhoatmojo, Satrio Baskoro. "Software & Software Engineering" IKI30202 - Rekayasa Perangkat Lunak Term 1 - 2011/2012. Faculty of Computer Science University of Indonesia. 2012. Print

- [Miles & Hamilton, 2006] Miles, Russ, and Kim Hamilton. Learning UML 2.0. Beijing: O'Reilly, 2006. Print.

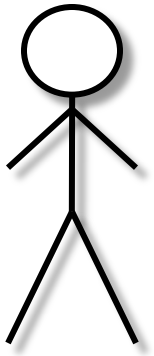
Tentang Use Case

- *Use case* adalah set aktivitas yang memproduksi beberapa hasil output.
- Mendeskripsikan bagaimana **sistem bereaksi terhadap aksi yang dilakukan aktor**.
- Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem.
- Menggambarkan kebutuhan sistem dari sudut pandang *user.*
- Menggambarkan hubungan antara *use case* dan aktor.
- *Use case* menggambarkan proses sistem (kebutuhan sistem dari sudut pandang *user*)
- *Use case* dibuat berdasarkan keperluan aktor →"apa" yang dikerjakan sistem, bukan "bagaimana" sistem mengerjakannya.

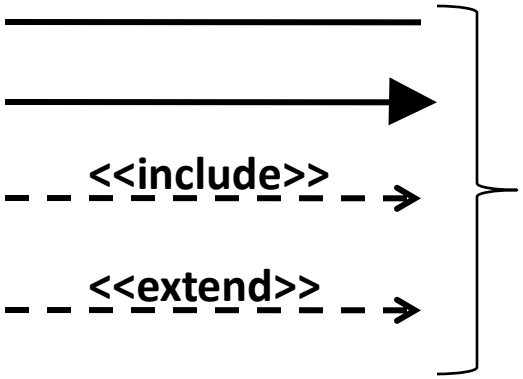
Manfaat Use Case

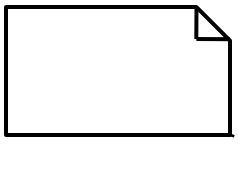
- Menggambarkan hasil *requirement* fungsional dalam bentuk diagram yang terstruktur.
- Melihat sistem per-fungsinya.
- Membantu menggambarkan komunikasi antara *user* dengan fungsi sistem dalam diagram yang mudah dimengerti.
- Membantu mengukur besarnya proyek, besarnya usaha, dan estimasi jadwal proyek.
- *Use case* diagram digunakan untuk mengambarkan interaksi antara pengguna sistem (aktor) dengan kasus (*use case*) yang disesuaikan dengan langkah-langkah (skenario) yang telah ditentukan. Sejak tahun 1992, dengan adanya pengembang UML, *use case* menjadi model utama (Requeirment Model) pada UML.

Komponen Use Case Diagram

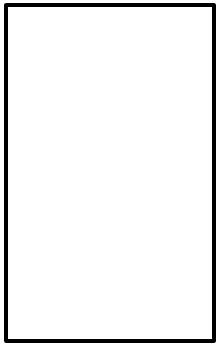


Aktor





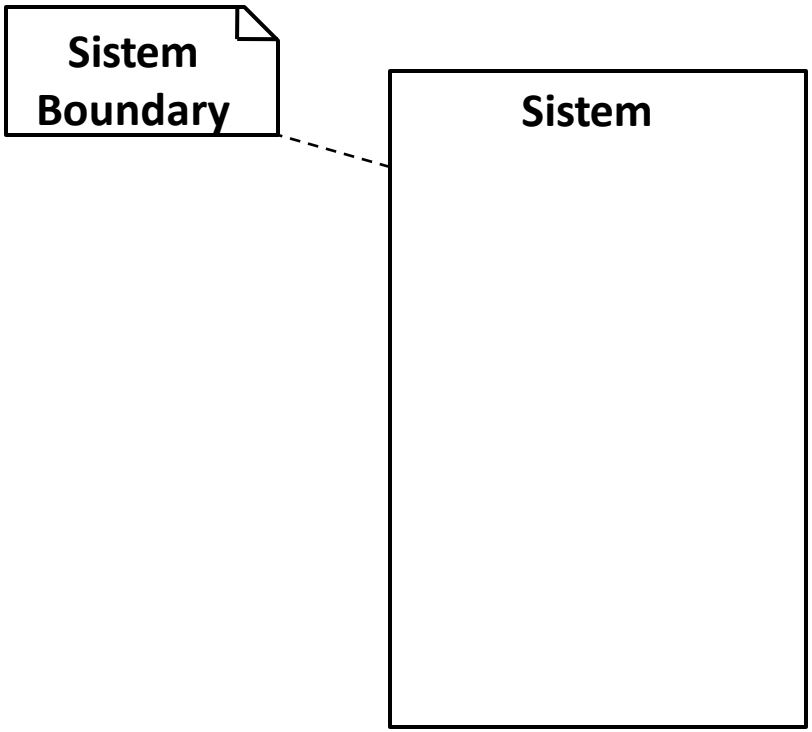
Catatan



System Boundary
(Batasan Sistem)

Sistem Boundary/Batasan Sistem

- Untuk mengelompokkan bagian di dalam sistem dengan di luar sistem.
- Tidak harus digunakan.



Aktor

- Menggambarkan **orang, sistem atau pihak luar** yang menyediakan atau menerima informasi dari sistem utama.
- Menggambarkan sebuah **tugas/peran** dan **bukannya posisi sebuah jabatan.**
- **Memberi input atau menerima informasi dari sistem.**
- Biasanya menggunakan **kata benda.**
- **Tidak boleh** ada komunikasi langsung antar aktor.
- Indikasi **<<system>>** pada aktor menerangkan sebuah sistem luar yang menjadi aktor.
- Terdapat aktor bernama “Time” yang mengindikasikan suatu kejadian yang terjadi secara periodik/bulanan (*scheduled events*).

3 Jenis Aktor

- **User/pengguna dari sistem**

  Biasanya merupakan orang secara fisik atau seorang *user*. Jenis ini merupakan yang paling umum dan ada disetiap sistem. **Contoh**: dalam sistem reservasi penerbangan, aktor merupakan orang yang secara langsung menggunakan sistem.

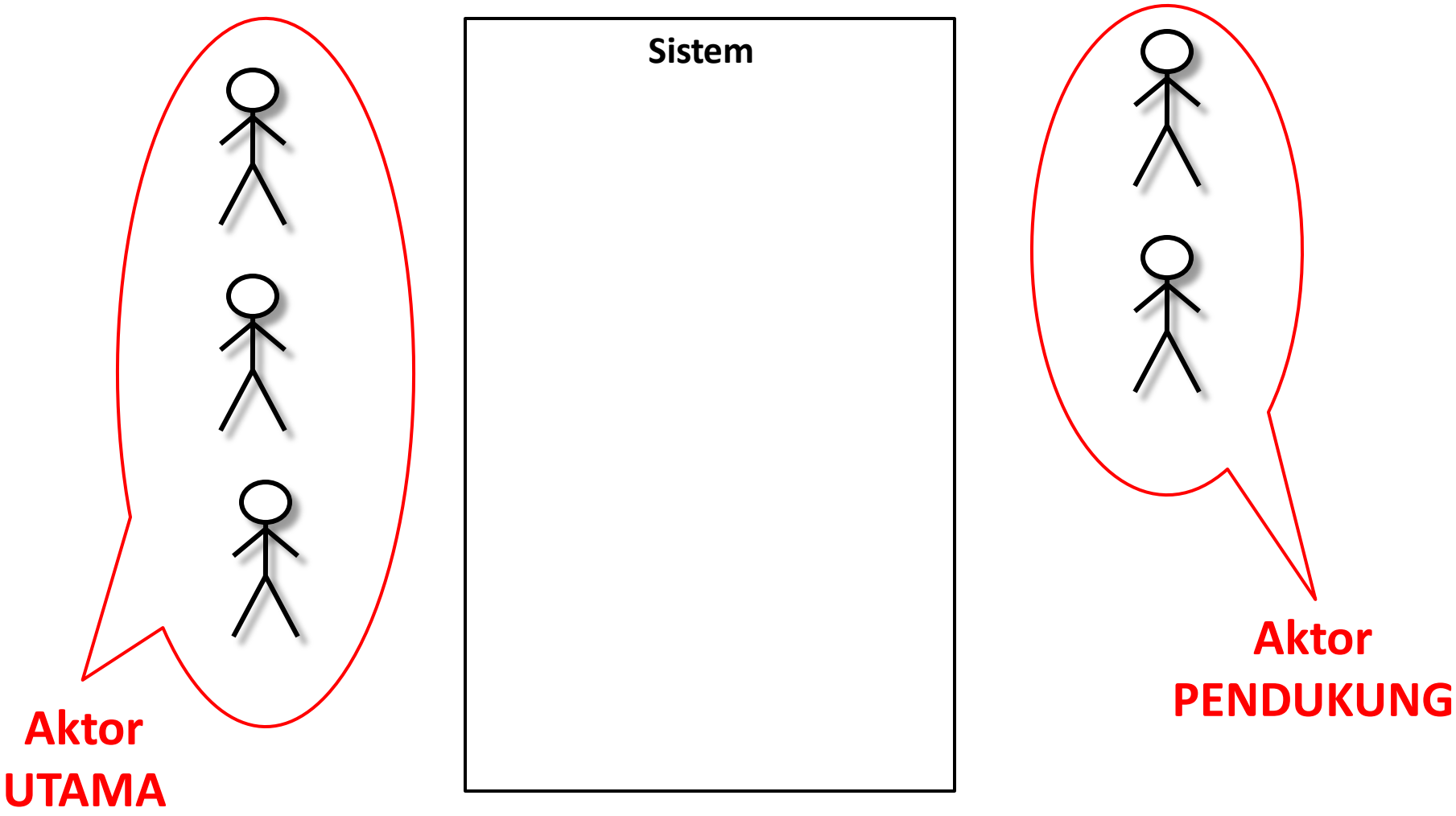
- **Sistem lain yang berinteraksi dengan sistem yang dibangun**

  Jenis aktor yang kedua adalah sistem lain. **Contoh**: sistem reservasi pesawat mungkin perlu interface dengan aplikasi eksternal untuk memvalidasi kartu kredit pembayaran. Dalam contoh ini, aplikasi kredit eksternal adalah sebuah aktor.

- **Waktu**

  Waktu menjadi sebuah aktor pada saat pada waktu tertentu memicu beberapa aksi dalam sistem. **Contoh**: sebagai bagian dari promosi diberikan kesempatan untuk memenangkan tiket gratis. Setiap hari pada jam 3 siang, sistem secara otomatis memilih secara acak konsumen untuk mendapatkan tiket gratis.

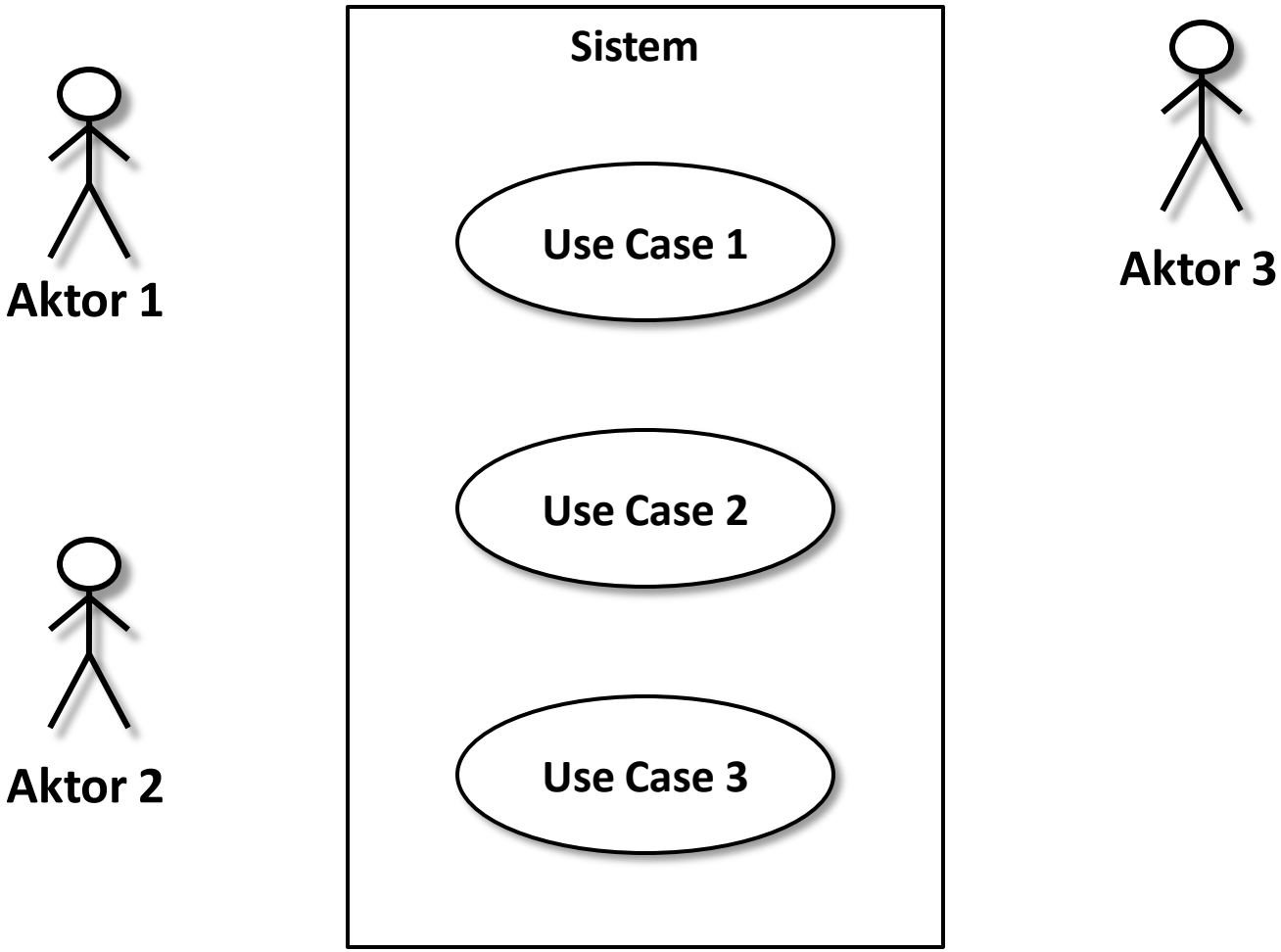
Penempatan Aktor



Penulisan Use Case

- *Use case* diberi nama dengan menyatakan apa hal yang dicapai dari hasil interaksinya dengan aktor.
- *Use case* dinotasikan dengan gambar elips.
- *Use case* biasanya dimulai dengan kata kerja.
- Penulisan nama *use case* biasanya di dalam gambar elips, namun boleh juga ditulis di atas atau di bawah gambar elips.
- Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada 2 use case yang memiliki nama yang sama.

A Sample of Use-Case Model Diagram



Relasi pada Use Case

Relasi pada Use Case

- Terdapat 4 jenis relasi yang mungkin terjadi pada *use case diagram*:
  1. Asosiasi antara aktor dan *use case*
  2. Asosiasi antar *use case*
  3. Generalisasi/Inheritance antar aktor
  4. Generalisasi/Inheritance antar *use case*

Asosiasi….

- Asosiasi = gabungan
- Asosiasi bukan menggambarkan aliran data/informasi.
- Asosiasi digunakan untuk menggambarkan bagaimana aktor terlibat dalam use case.

Asosiasi antara Aktor – Use Case

- **Gunakan garis tanpa panah** untuk asosiasi antara aktor dan *use case*.

Asosiasi antara Aktor – Use Case

- **Contoh**: kustomer melakukan pembayaran tiket dan memilih pembayaran dengan kartu kredit, kemudian sistem melakukan validasi kartu ke sistem kartu kredit.

Asosiasi antar Use Case

- Ada 2 jenis relasi asosiasi (*association relationship*) antar sesama *use case,* yaitu:
  - **Relasi “Includes”**
  - **Relasi “Extends”**

Relasi “Includes”

- Relasi “includes” mengizinkan sebuah *use case* untuk **menggunakan fungsi** yang disediakan oleh *use case* lain. Relasi ini dapat digunakan pada kasus sbb:

- **Pertama:** jika terdapat dua atau lebih *use case* yang memiliki **sebuah fungsi besar yang identik**, yang **selalu** dilakukan jika suatu *use lain* dilakukan. Kemudian masing-masing *use case* dapat memiliki sebuah relasi “includes” dengan *use case* ini.

- **Kedua:** Jika sebuah *use case* tunggal memiliki sejumlah besar fungsi yang tidak biasa.

Relasi “Includes”

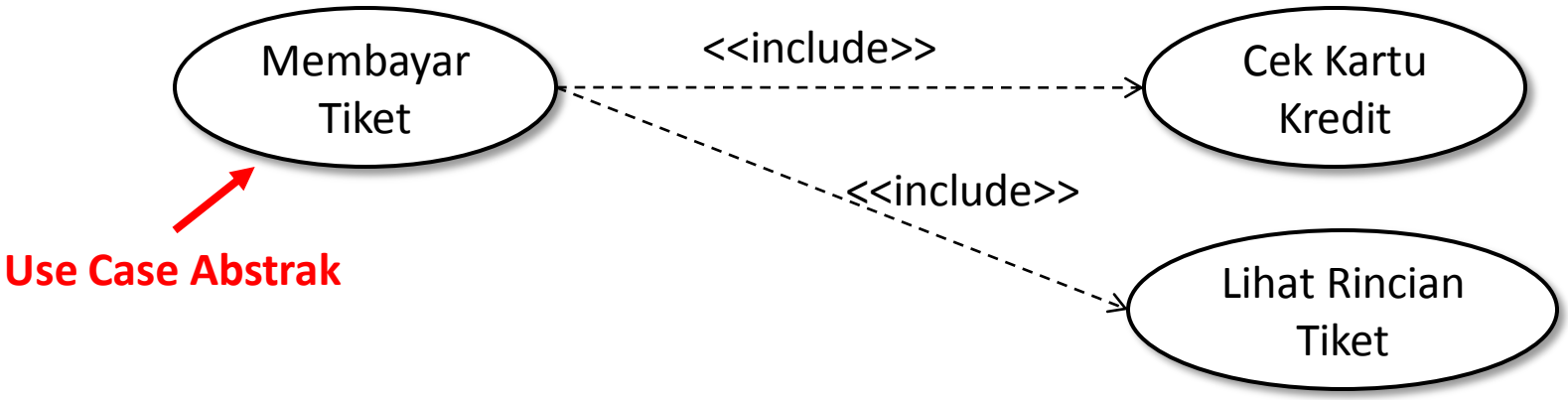
- *Use case* yang dibuat dari fungsi identik beberapa *use case* lain disebut dengan ***use case* abstrak**.

- ***Use case* abstrak** tersedia bagi *use case* lain yang membutuhkan fungsionalitas *use case* abstrak tersebut.
- Relasi antara *use case* abstrak dengan dengan *use cese* yang menggunakannya **disebut relasi “includes”**.
- Relasi ini digambarkan dengan panah dari *use case* abstrak ke *use case* yang menggunakannya, dan diberi label <<include>>.

Relasi "Includes"

Contoh:

Setiap melakukan pembayaran tiket, **selama proses tersebut** dilakukan juga pengecekan kartu kredit dan menampilkan rincian tiket.



Relasi “Extends”

- **Use case extension**
  - *Use case* yang memiliki fungsi tambahan yang dilakukan pada *use case* lain.
  - *Use case* yang memperluas fungsi dari *use case* aslinya.

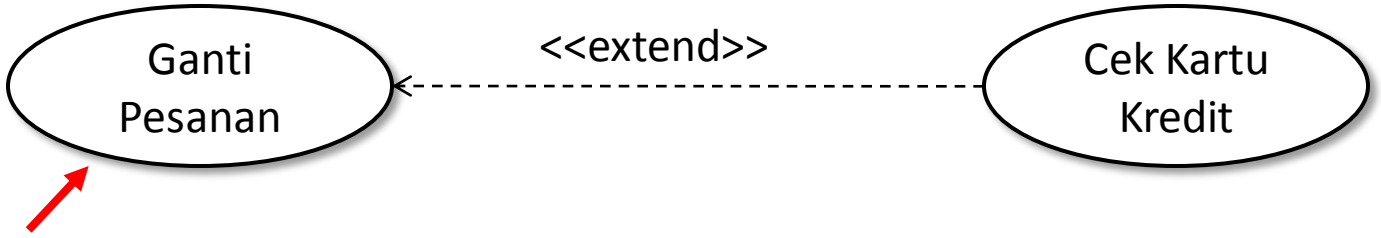
- Relasi “extend” digambarkan dengan garis putus-putus dengan ujung panah mengarah ke *use case* aslinya, dan diberi label <<extend>>.

Relasi "Extends"

Contoh:

Setiap melakukan ganti pesanan, lakukan cek kartu kredit **jika dan hanya jika** total harga pesanan berubah. Jika setelah ganti pesanan harga total pesanan tidak berubah, maka tidak perlu melakukan cek kartu kredit.

Karena "Cek Kartu Kredit" tidak selalu dilakukan, maka ada sebuah relasi "extend" dari *use case* yang secara opsional berjalan ("Cek Kartu Kredit") ke use case yang di-*extend* ("Ganti Pesanan").



Sudah jelas perbedaan antara relasi “include” dan relasi “extends”?

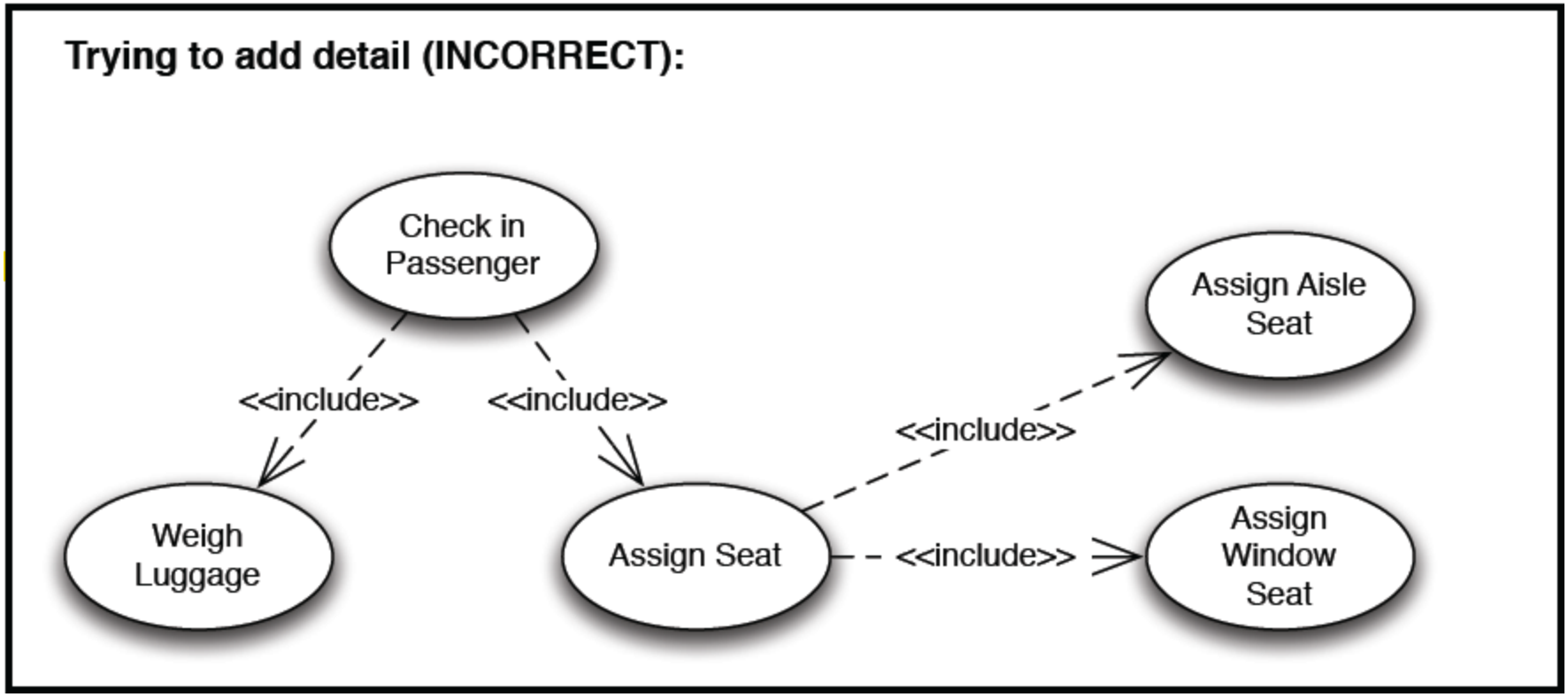
Apa Perbedaan antara Includes dan Extends?

- “A includes B" artinya: tugas “A” adalah subtugas dari “B”; dalam menyelesaikan tugas “A”, tugas “B” akan diselesaikan paling tidak sekali.

- “A extends B" artinya: “A” adalah tugas yang bertipe sama dengan “B”, namun “A” spesial, kasusnya lebih spesifik dari “B”. Jadi, “A” mirip dengan “B”, namun “A” memiliki proses ekstra yang harus dilakukan dalam menyelesaikan “B”.

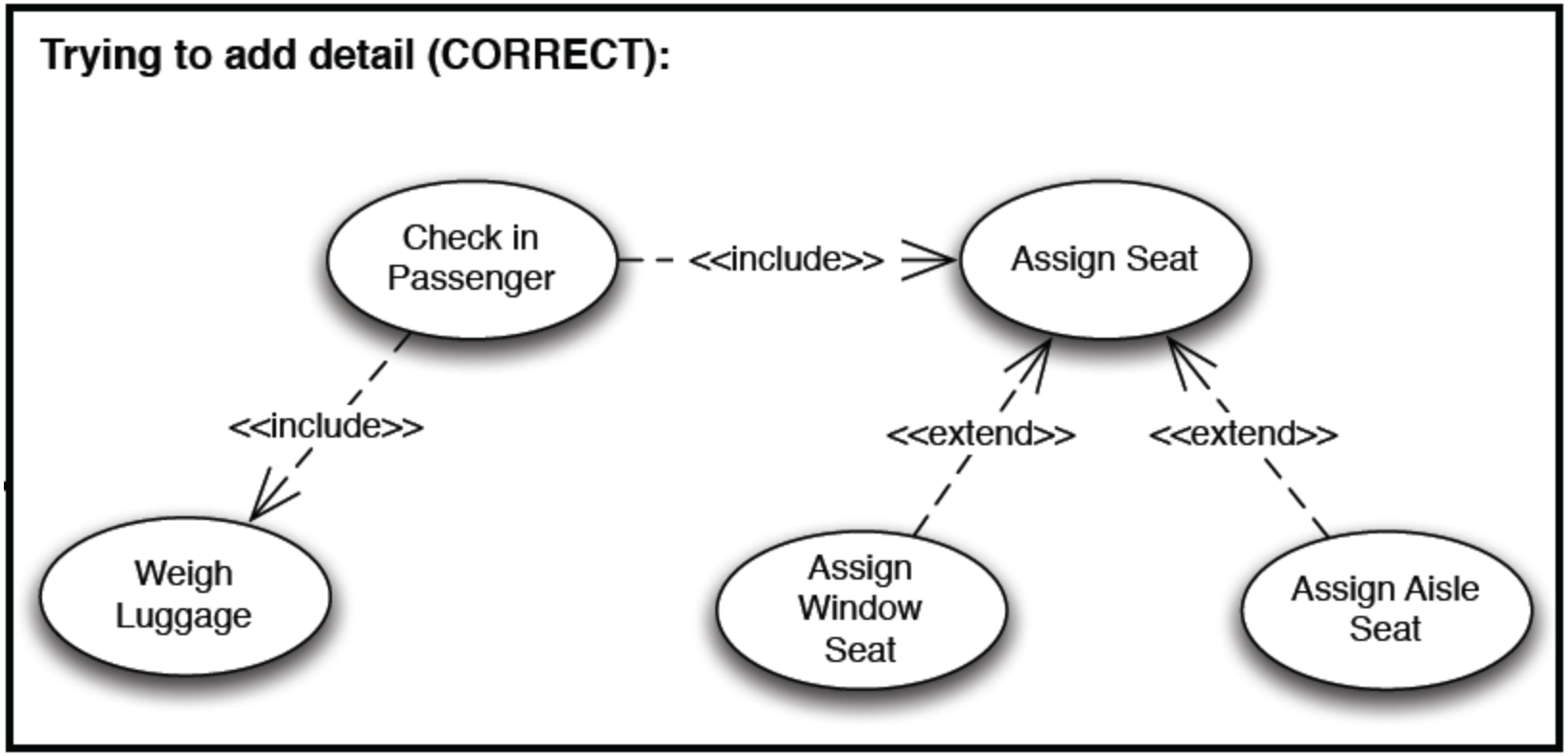
Contoh... (salah) ☹

- Diagram ini mengatakan bahwa dalam memesan kursi pesawat **kamu harus memesan kursi penumpang bagian jandela dan bagian tengah sekaligus.**



Contoh... (benar) ☺

- Diagram ini mengatakan bahwa **ada dua pilihan** dalam memesan kursi penumpang, pesan kursi di pinggir jendela atau kursi tengah, namun pemesan **hanya memilih satu dari dua kursi tersebut**.



Generalisasi antar Aktor

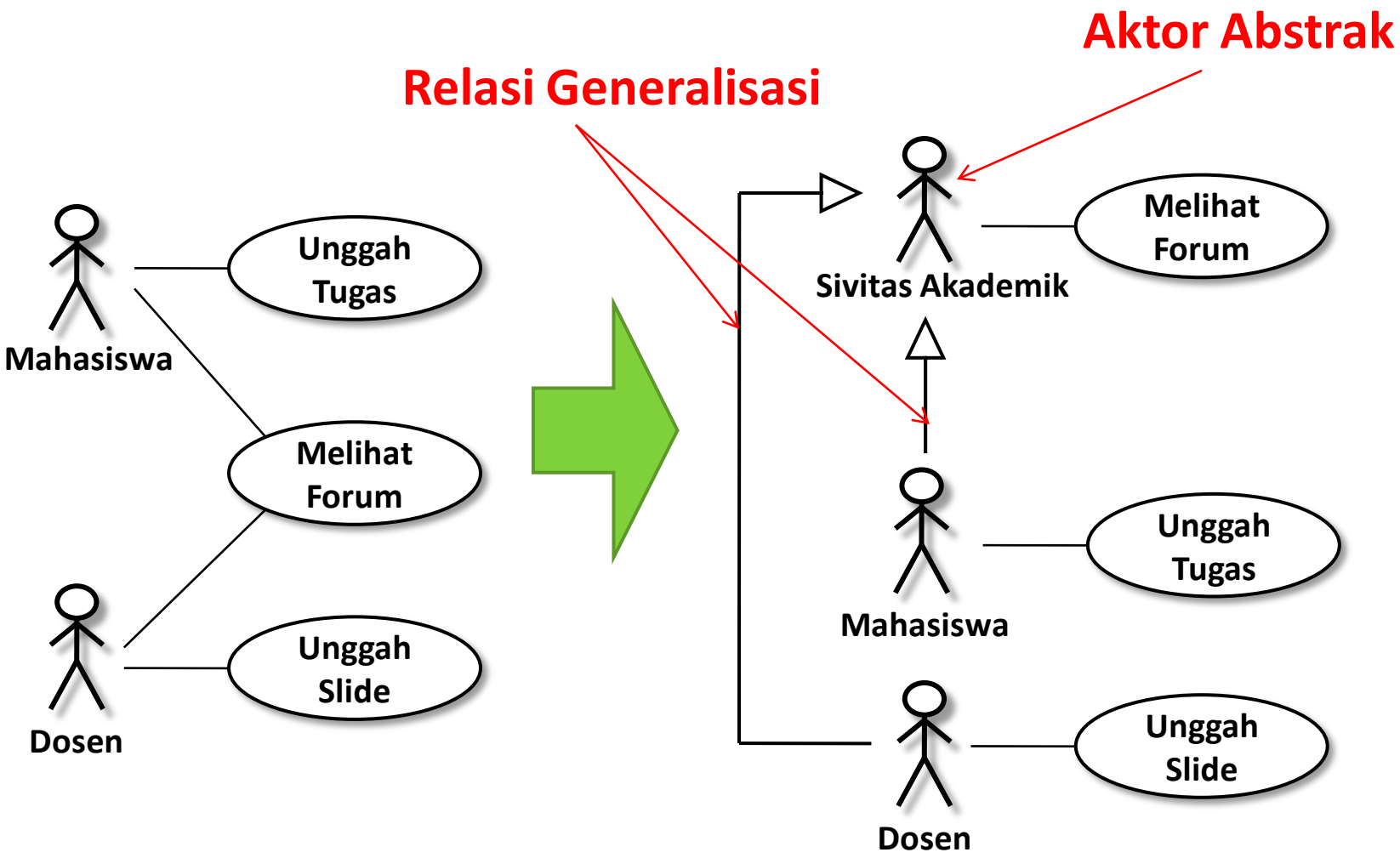
- **Aktor generalisasi = Aktor inheritance (pewarisan)**

- **Aktor generalisasi** - beberapa aktor yang memiliki beberapa persamaan dan melakukan *use case* yang sama.
- Beberapa aktor yang mewarisi interaksi aktor abstrak terhadap sistem.
- Digambarkan dengan garis dengan panah tertutup yang mengarah ke aktor abstrak.

Generalisasi antar Aktor

- Contoh:
  - Aktor dosen dan aktor mahasiswa sama-sama diizinkan untuk melihat forum.
  - Dalam hal ini dibuatkan satu aktor generalisasi (aktor abstrak) yang akan menuju ke *use case* “Melihat Forum”.
  - Aktor dosen dan aktor mahasiswa sama-sama mewarisi aktor abstrak tersebut.
  - Contoh *use case* diagramnya… →

Generalisasi antar Aktor



Generalisasi antar Use Case

- ***Use case* generalisasi = *Use case* inheritance (pewarisan)**

Dari [Miles & Hamilton, 2006]:

- Kondisi dimana satu *use case* merupakan tipe spesial dari *use case* yang lain.
- Langkah yang baik untuk menggunakan kembali *use case* sehingga analis **hanya butuh menspesifikasi langkah ekstranya** (spesifiknya) pada *use case* spesifik.

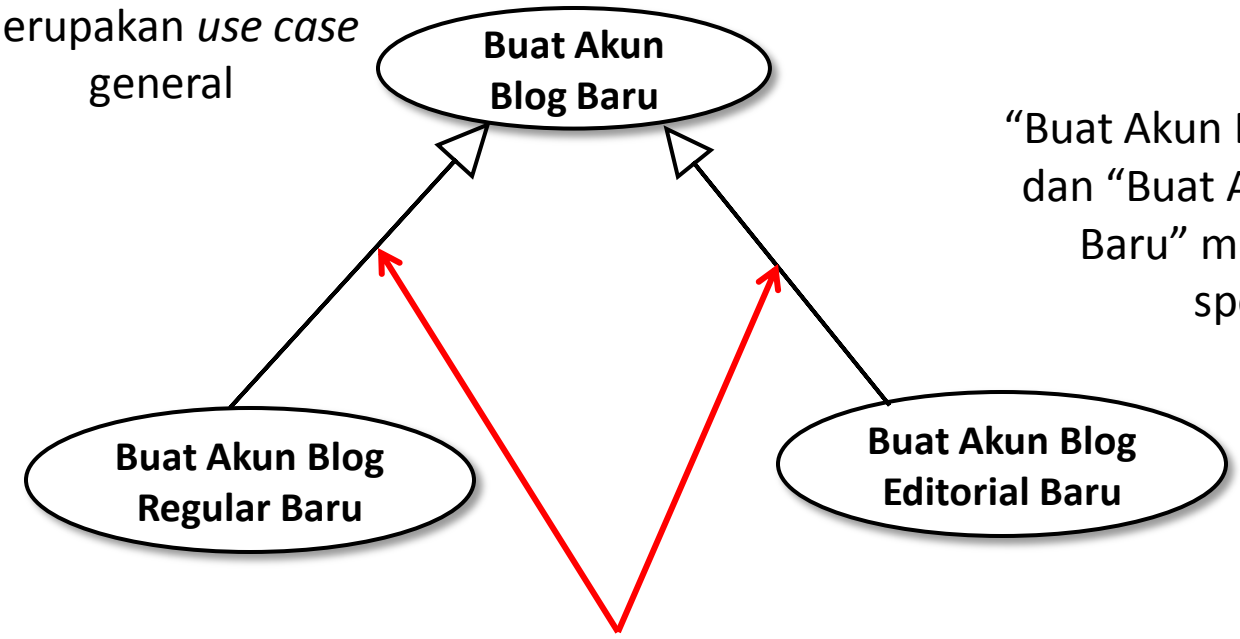
Generalisasi antar Use Case

- Hal yang harus diperhatikan saat membuat *use case* generalisasi:
  - **Setiap langkah di *use case* general harus** terjadi di *use case* spesial.
  - **Setiap relasi dari *use case* general** dengan aktor juga harus sesuai dengan *use case* spesial.
- **Jika tidak ingin *use case* spesial** melakukan apapun yang dilakukan oleh *use case* general, maka **jangan gunakan generalisasi *use* cae.**

Generalisasi antar Use Case

Contoh: Pada saat membuat blog akun baru, ada dua tipe blog yang tersedia, blog regular dan blog editorial. Kedua blog ini memiliki spesifikasi berbeda, namun secara umum sama-sama membuat suatu blog akun baru.

"Buat Akun Blog Baru" merupakan *use case* general



"Buat Akun Blog Regular Baru" dan "Buat Akun Blog Editorial Baru" menjadi *use case* spesifiknya.

Sudah SIAP Untuk Membuat USE CASE DIAGRAM?

Mari kita latihan lagi … →

Contoh Kasus 1

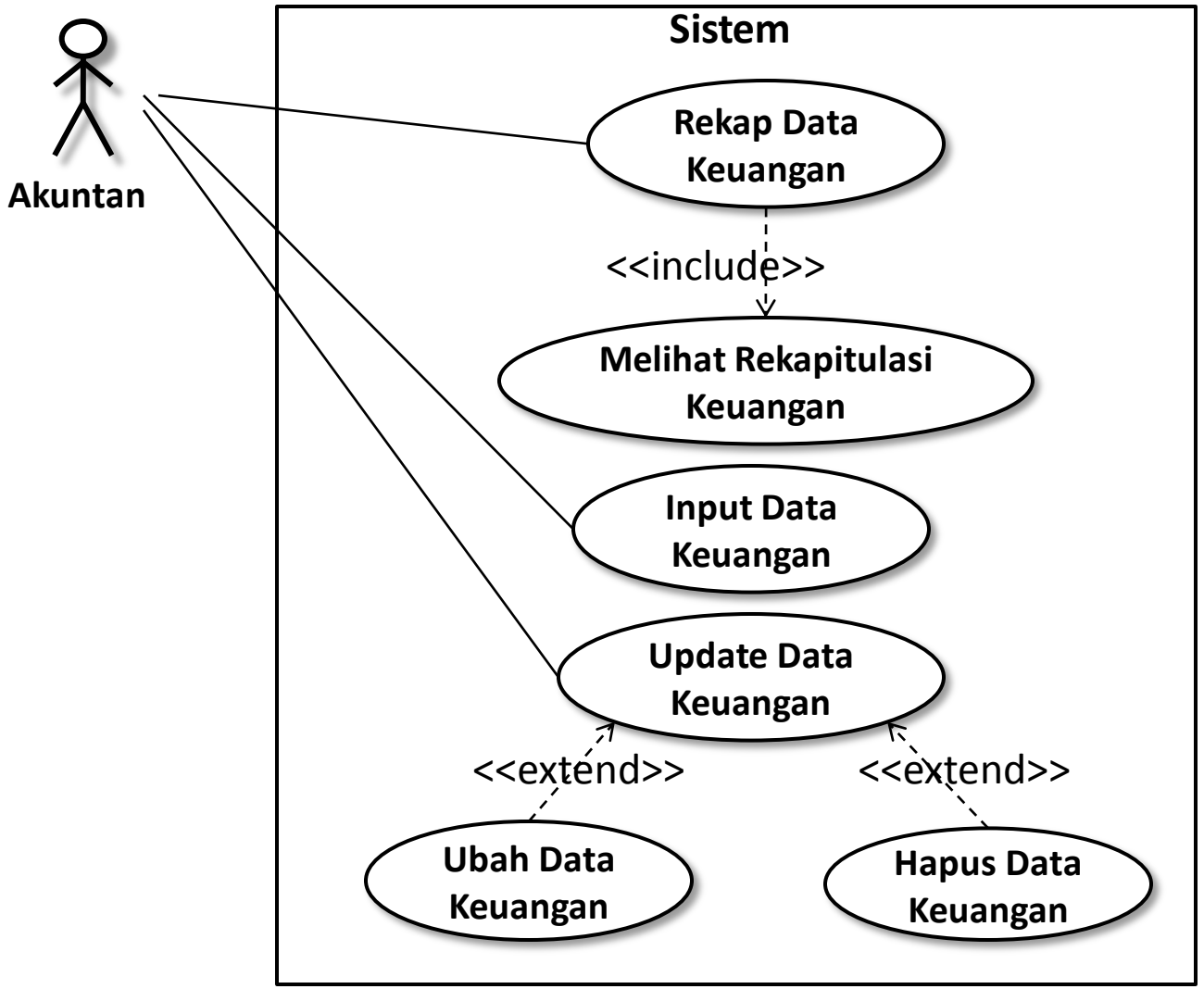
Sistem Informasi Rekapitulasi Keuangan (SIReKe)

- Sebuah perusahaan yang sedang berkembang ingin membuat aplikasi rekapitulasi keuangan perusahaan untuk melakukan proses pembukuan.
- Akuntan dapat melakukan input data keuangan, ubah data keuangan, dan hapus data keuangan pada sistem ini.
- Untuk membuat rekapitulasi keuangan, Akuntan harus masuk ke fitur melihat rekapitulasi keuangan terlebih dahulu, kemudian memasukkan input tanggal yang diinginkan dan program rekapitulasi akan secara otomatis membuat laporan pembukuan dari input akuntan.

Contoh Kasus 1

- **Aktor:**
  - Akuntan
    - Input data keuangan
    - Ubah data keuangan, termasuk menghapusnya.
    - Rekap Data Keuangan
    - Melihat Rekapitulasi Keuangan

Contoh Kasus 1



Contoh Kasus 2

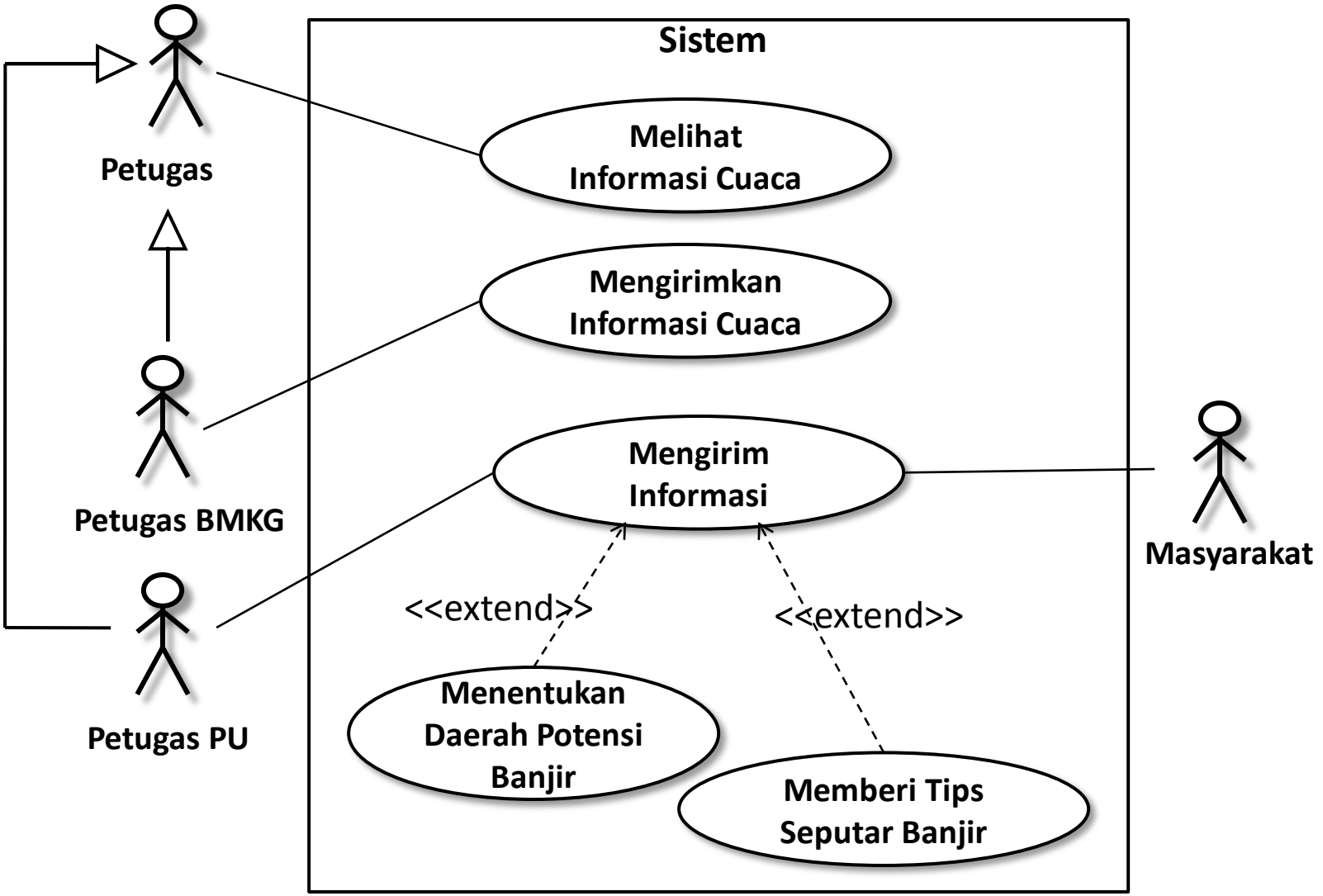
Sistem Waspada Banjir (SiWaJir)

- BMKG ingin membuat suatu sistem yang dengan segera dapat memberi peringatan dini bencana banjir ke pada masyarakat.
  - Petugas BMKG mengirimkan informasi cuaca ke dalam sistem. Informasi cuaca hanya dapat dilihat oleh Petugas PU dan petugas BMKG sendiri.
  - Petugas PU Memberikan update informasi geografis banjir, yaitu menentukan daerah berpotensi banjir dan tips seputar banjir.
  - Masyarakat menerima semua informasi yang dikirimkan oleh pertugas PU.

Contoh Kasus 2

- **Aktor:**
  - Petugas BMKG
    - Mengirimkan informasi cuaca
    - Melihat informasi cuaca
  - Petugas PU
    - Melihat informasi cuaca
    - Memberikan update informasi geografis banjir, yaitu menentukan daerah berpotensi banjir dan tips seputar banjir.
  - Masyarakat
    - Melihat informasi daerah potensi banjir dan tips seputar banjir.

Contoh Kasus 2



Pertanyaan?