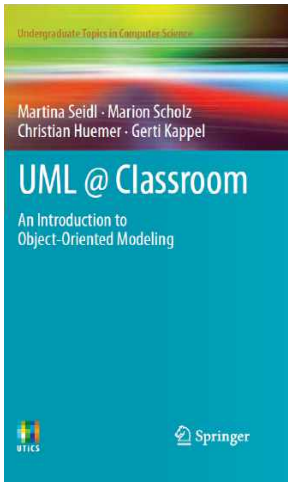
Object-Oriented Modeling

Activity Diagram

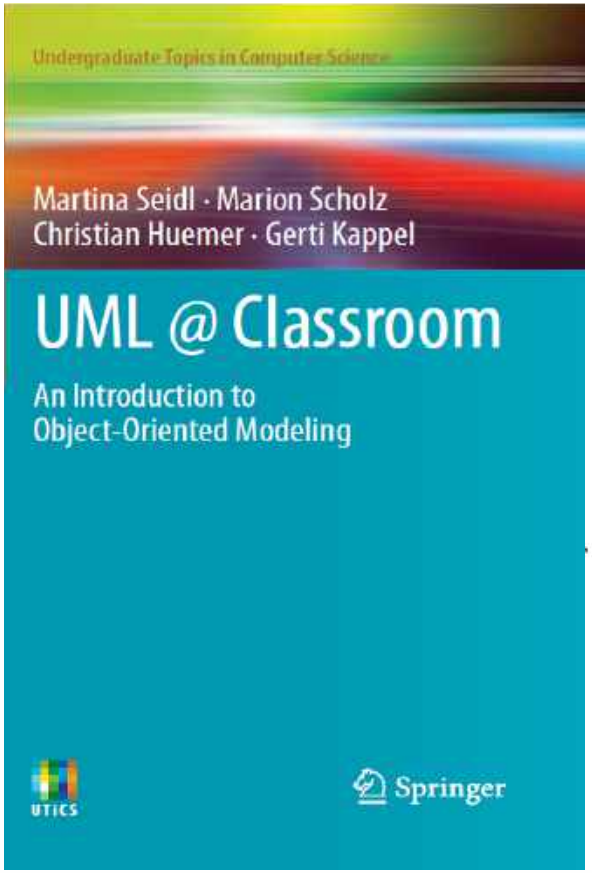


Slide untuk melengkapi buku UML@Classroom
Versi 1.0
Diterjemahkan dari
slide milik Busines Informatics Group, Vienna University of Technology

Program Studi Teknik Informatika
Jurusan Teknik Informatika
Politeknik Negeri Batam
Jalan Ahmad Yani, Batam Center, Batam 29461
www.polibatam.ac.id

Pustaka

- Materi kuliah diambil dari buku berikut:



UML @ Classroom:
An Introduction to Object-Oriented Modeling
Martina Seidl, Marion Scholz, Christian Huemer and Gerti Kappel

Springer Publishing, 2015

ISBN 3319127411

- Use Case Diagram
- Structure Modeling
- State Machine Diagram
- Sequence Diagram
- **Activity Diagram**

Materi

- Pengenalan
- Activity
- Action
- Edge
  - Control flow
  - Object flow
- Initial node, activity final node, flow final node
- Alternative path
- Concurrent path
- Object node
- Event-based action dan call behavior action
- Partition
- Exception handling

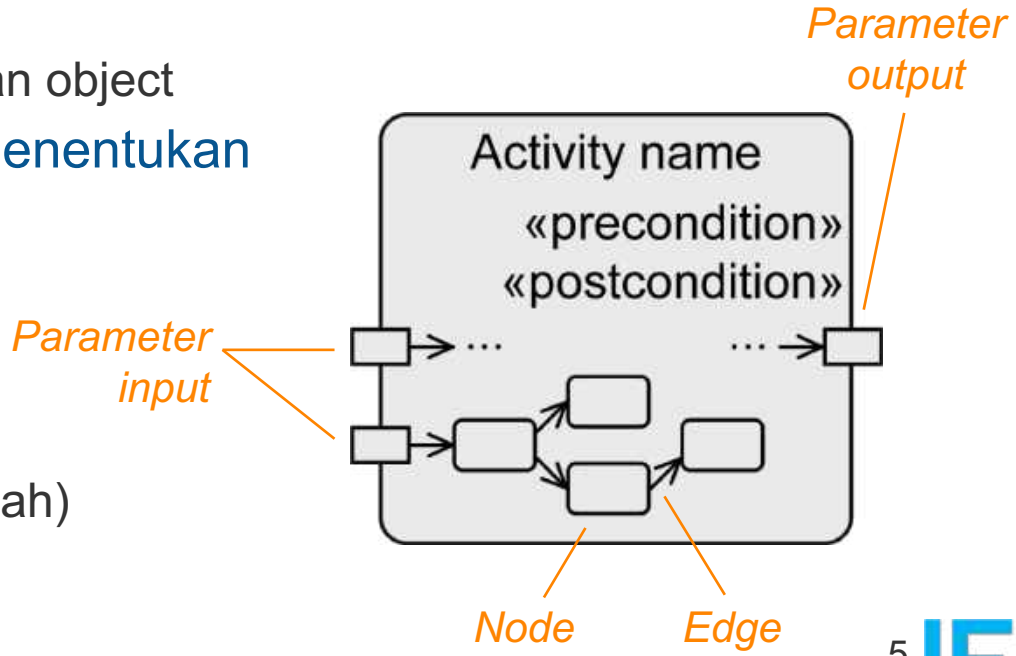
Pengenalan

- Fokus activity diagram: **aspek pemrosesan secara prosedural**
- Konsep bahasa berorientasi aliran
- Berdasar pada
  - bahasa untuk mendefinisikan proses bisnis
  - konsep untuk mendeskripsikan proses komunikasi konkuren (konsep token yang ditemukan pada petri net)
- Konsep dan notasi mencakup **aplikasi dalam lingkup yang luas**
  - Memodelkan sistem berorientasi objek maupun yang tidak berorientasi objek

Activity

Activity A

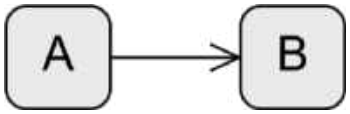
- Spesifikasi perilaku yang didefinisikan oleh user pada berbagai level
- Contoh:
  - Definisi perilaku sebuah operasi dalam bentuk instruksi
  - Memodelkan urutan langkah dalam sebuah use case
  - Memodelkan fungsi sebuah proses bisnis
- Sebuah activity adalah graph berarah/*directed graph*
  - Nodes: aksi dan activity
  - Edges: untuk aliran kontrol dan object
- Aliran kontrol dan object flow menentukan eksekusi
- Opsional:
  - Parameter
  - Pre- dan postconditions (keadaan sebelum dan sesudah)



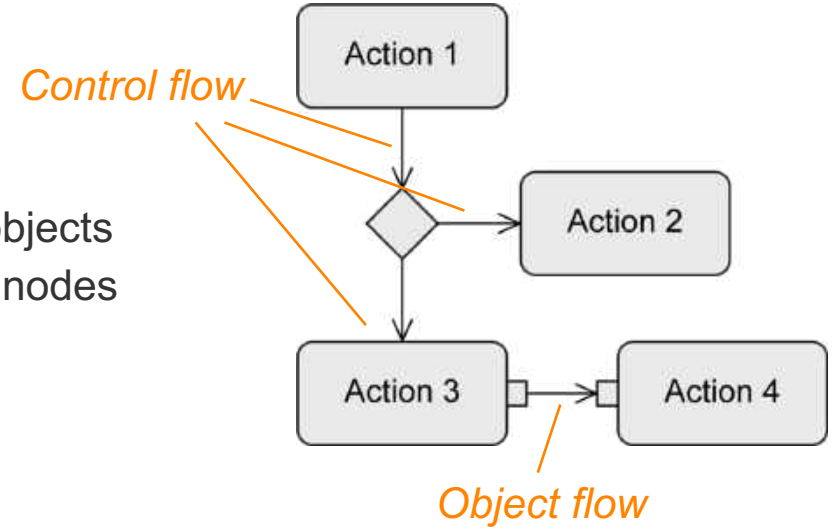
Action

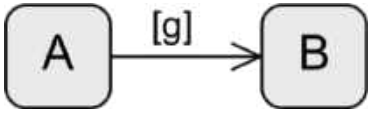
- **Elemen dasar** untuk menggambarkan perilaku yang didefinisikan oleh pengguna
- **Atomic** tapi dapat dihentikan
- Tidak ada aturan khusus untuk penggambaran sebuah action
  →Definisi dalam bahasa manusia atau bahasa pemrograman
- Memproses nilai input untuk menghasilkan nilai output
- Notasi khusus untuk berbagai tipe action, diantaranya yang paling penting adalah
  - Action berdasarkan event (*event-based actions*)
  - Action memanggil perilaku (*call behavior actions*)

Edges (sisi)



- Menghubungkan antar activity dan action
- Menggambarkan urutan eksekusi
- Tipe
  - *Control flow edges*
    - Menggambarkan urutan antar nodes
  - *Object flow edges*
    - Digunakan untuk pertukaran data atau objects
    - Menggambarkan kebergantungan antar nodes (data/causal)
- Guard (condition)
  - Flow control dan object hanya dapat berjalan jika guards dalam kurung siku bernilai benar

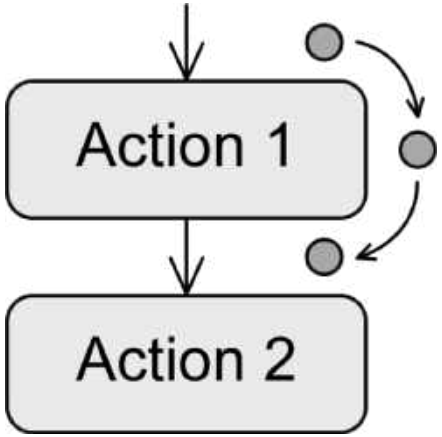




Token

- **Mekanisme koordinasi virtual** yang menggambarkan eksekusi yang sebenarnya
  - Tidak ada komponen fisik dalam diagram
  - Mekanisme yang memberikan ijin eksekusi kepada action

- Jika sebuah action menerima token, action tersebut dapat dijalankan
- Saat sebuah action selesai, action tersebut memberikan token kepada action berikutnya dan eksekusi action tersebut dimulai
- Guards dapat mencegah token diteruskan
  - Tokens are stored in previous node

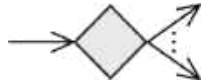


- Control token dan object token
  - **Control token**: “ijin eksekusi" untuk sebuah node
  - **Object token**: mengirim data + “ijin eksekusi”

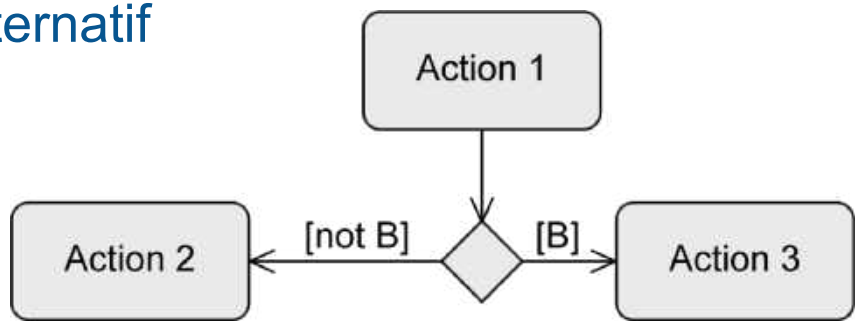
Awal dan Akhir Activity

- Initial node
  - Memulai eksekusi sebuah activity
  - Memberikan token ke semua edge keluar
  - Menyimpan token sampai semua node berikutnya menerimanya
  - Lebih dari satu initial node untuk menggambarkan concurrency
- Activity final node
  - Mengakhiri semua flow dalam sebuah activity
  - Token pertama yang mencapai activity final node menghentikan seluruh activity
    - Termasuk semua concurrent subpaths
  - Control dan object token lain dihapus
    - Perkecualian: object token yang sudah ada dalam output parameter activity
- Flow final node
  - Mengakhiri sebuah execution path dalam sebuah activity
  - Semua token lain dalam activity tidak terpengaruh

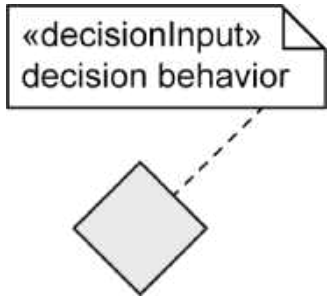
Alternative Path – Decision Node



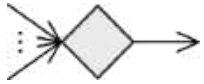
- Untuk menggambarkan cabang alternatif
- „Switch point“ untuk token
- Edge keluar memiliki guard
  - Syntax: [Boolean expression]
  - Token memilih **satu** cabang
  - Guard harus mutually exclusive (tidak boleh berpotongan)
  - Predefined: [else]



- Decision behavior
  - Menggambarkan perilaku yang diperlukan agar nilai guard dapat ditentukan
  - Eksekusinya tidak boleh memiliki efek samping

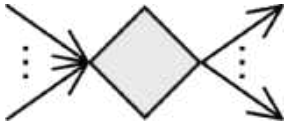


Alternative Paths – Merge Node

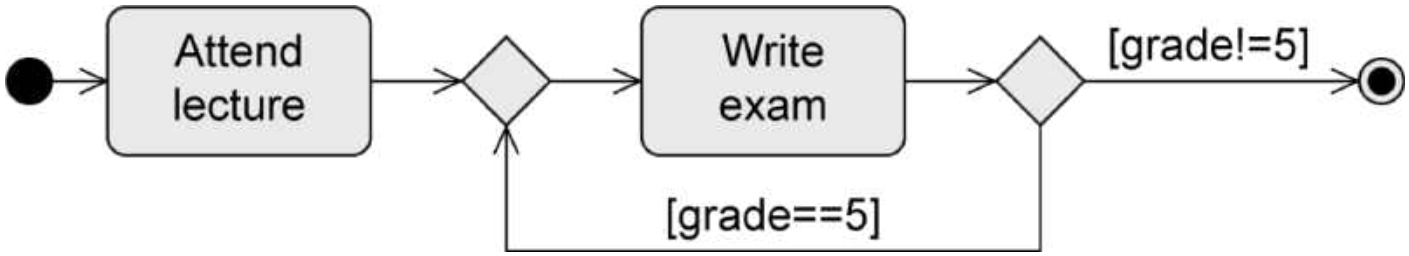


- Untuk menggabungkan **alternative** subpath
- Memberikan token node berikutnya

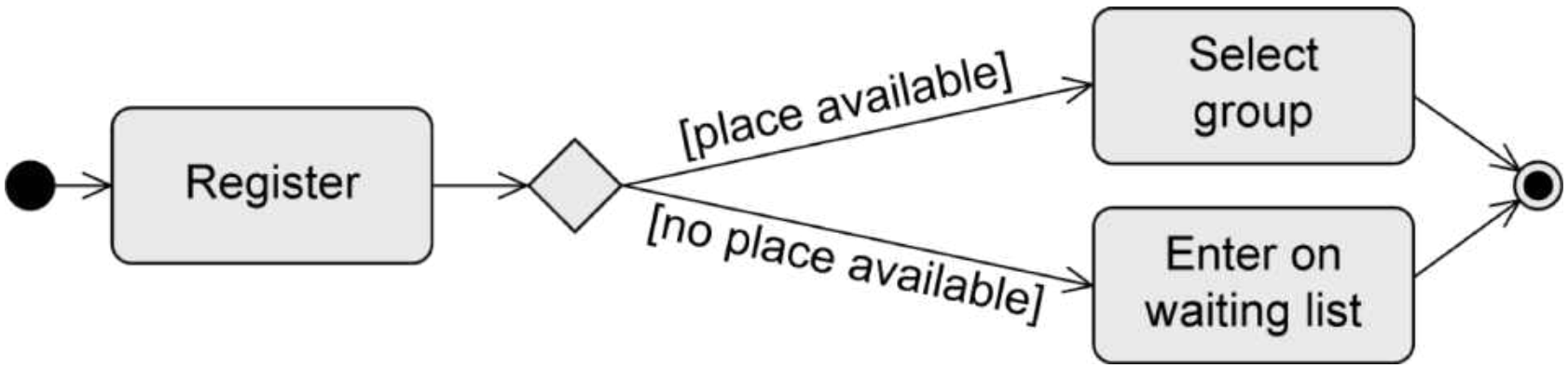
- Mengkombinasikan decision dan merge node



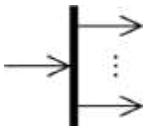
- Decision dan merge node dapat juga digunakan untuk menggambarkan pengulangan/loops:



Contoh: Alternative Paths

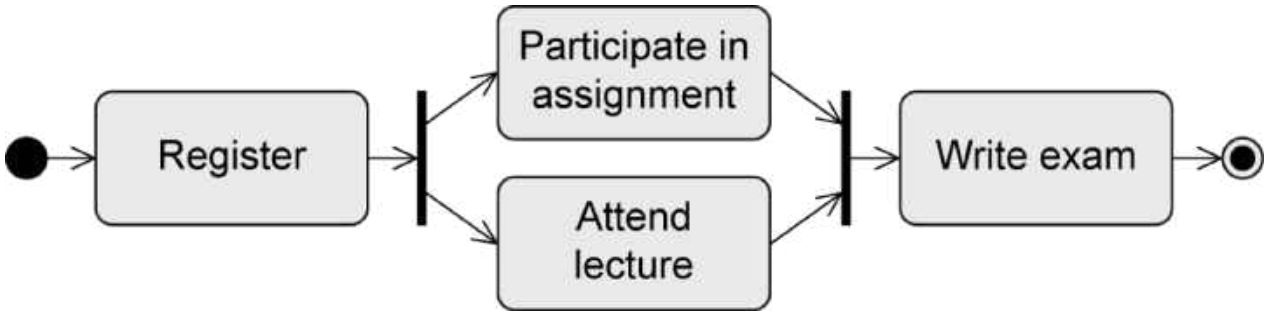


Concurrent Paths – Parallelization Node

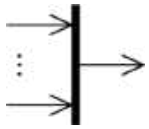


- Untuk memisahkan path menjadi subpath yang concurrent
- Menggandakan token untuk semua edge keluar

- Contoh:

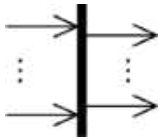


Concurrent Paths – Synchronization Node

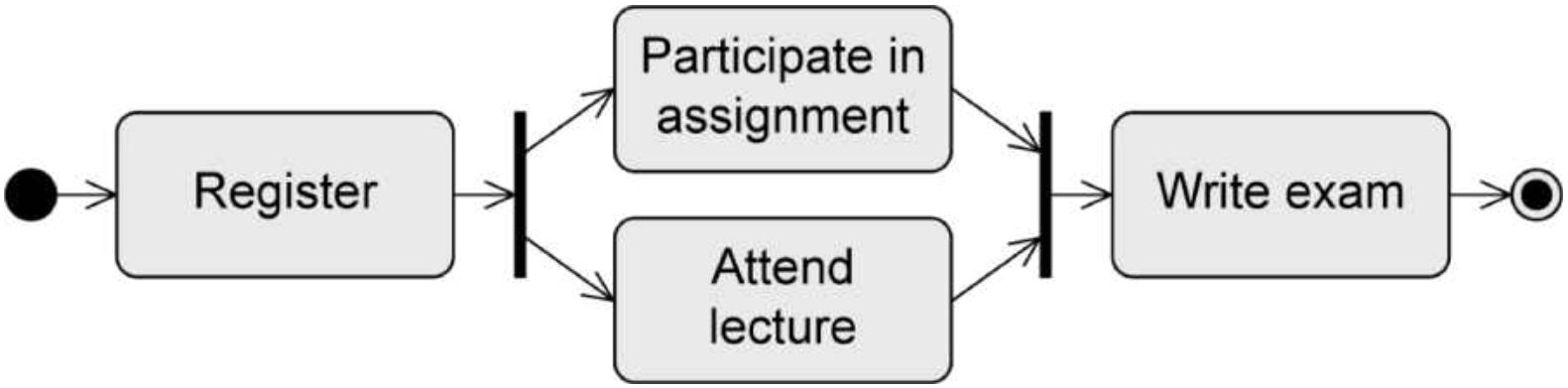


- Untuk menggabungkan subpaths yang concurrent
- Pemrosesan token
  - Menunggu sampai semua token dari semua edge masuk (*ingoing edge*)
  - Menggabungkan semua control token menjadi satu token dan meneruskannya
  - Meneruskan semua object token

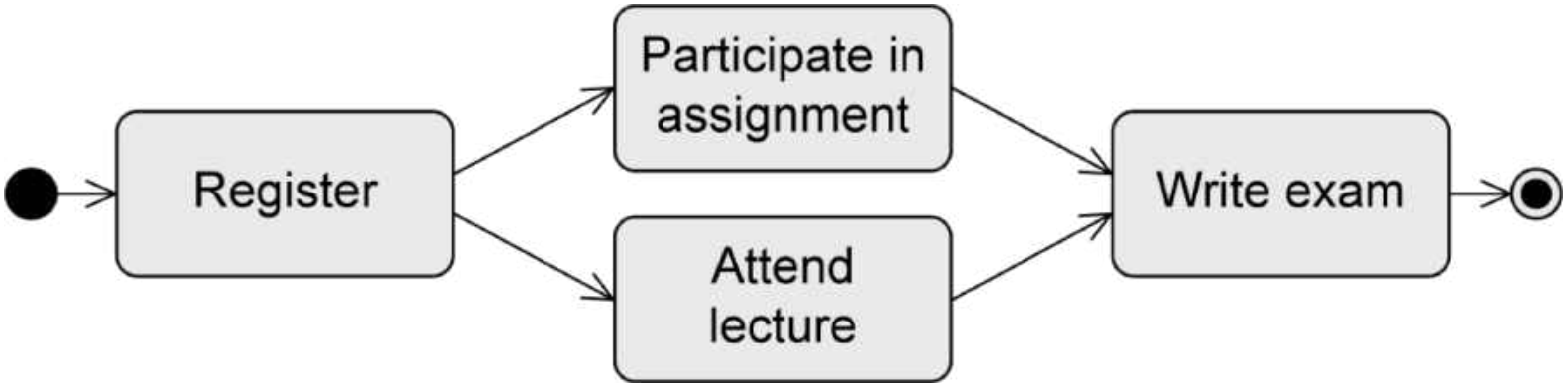
- Mengkombinasikan parallelization dan synchronization node:



Contoh: Equivalent Control Flow

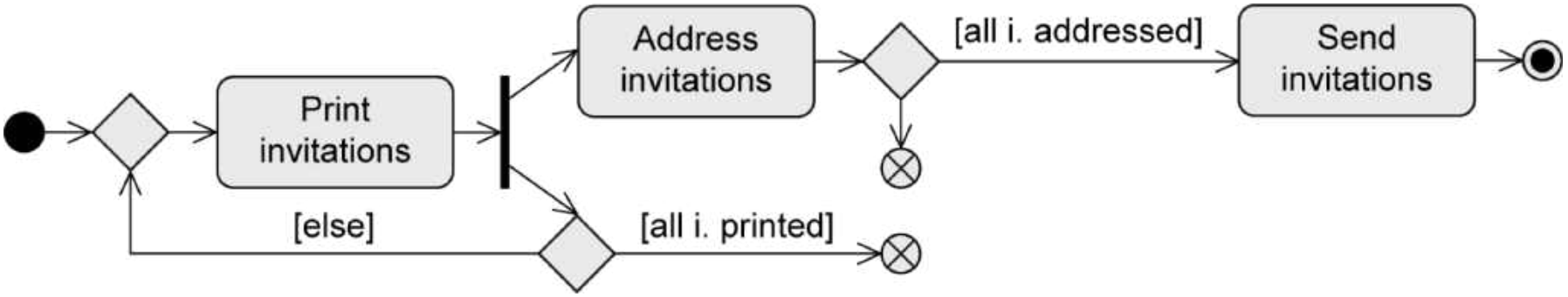


… sama dengan …

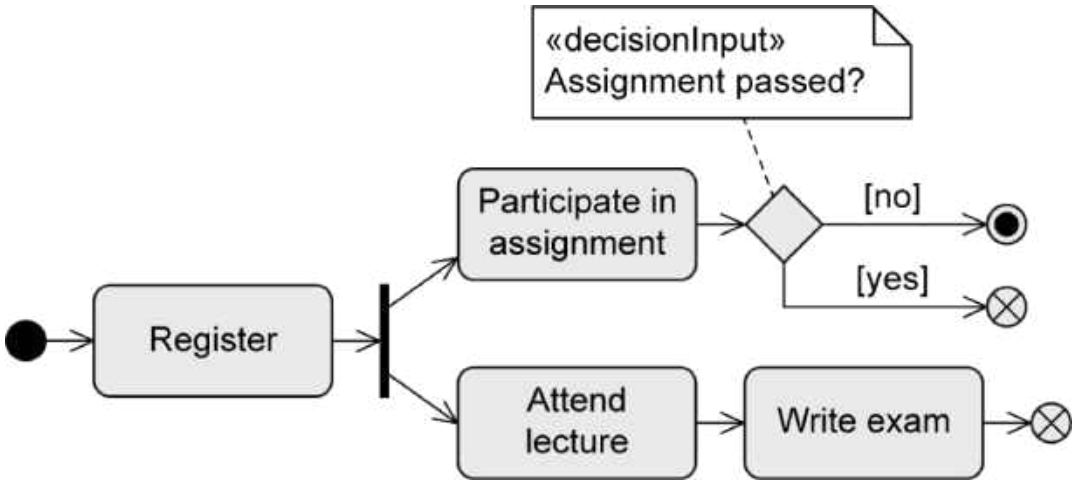


Contoh: Membuat dan Mengirim Undangan Meeting

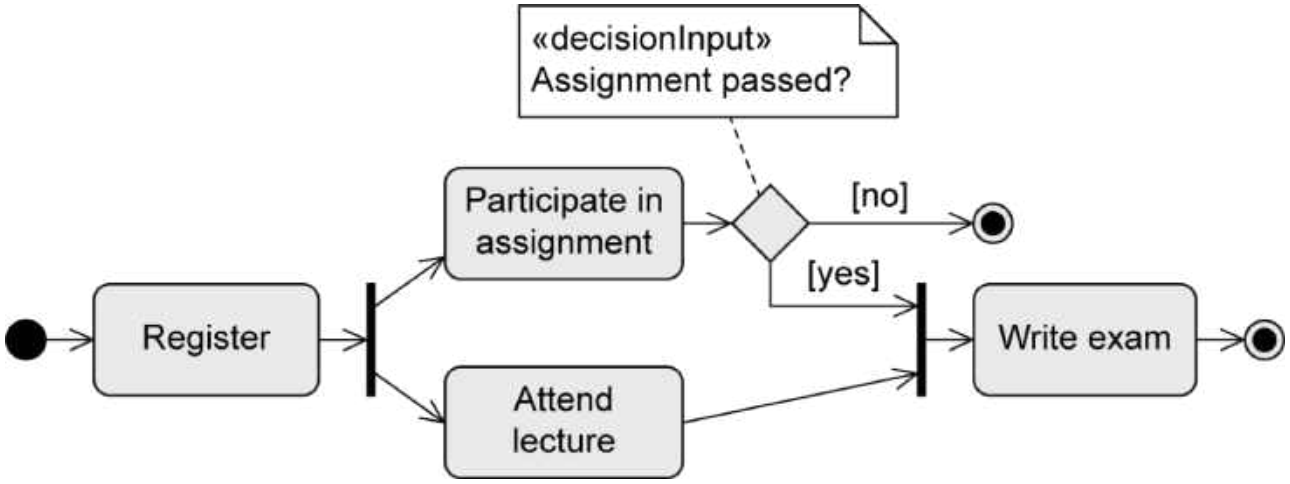
- Saat undangan dicetak, undangan yang telah dicetak diberi alamat pengiriman.
- Saat seluruh undangan telah diberi alamat, maka undangan dapat dikirimkan.



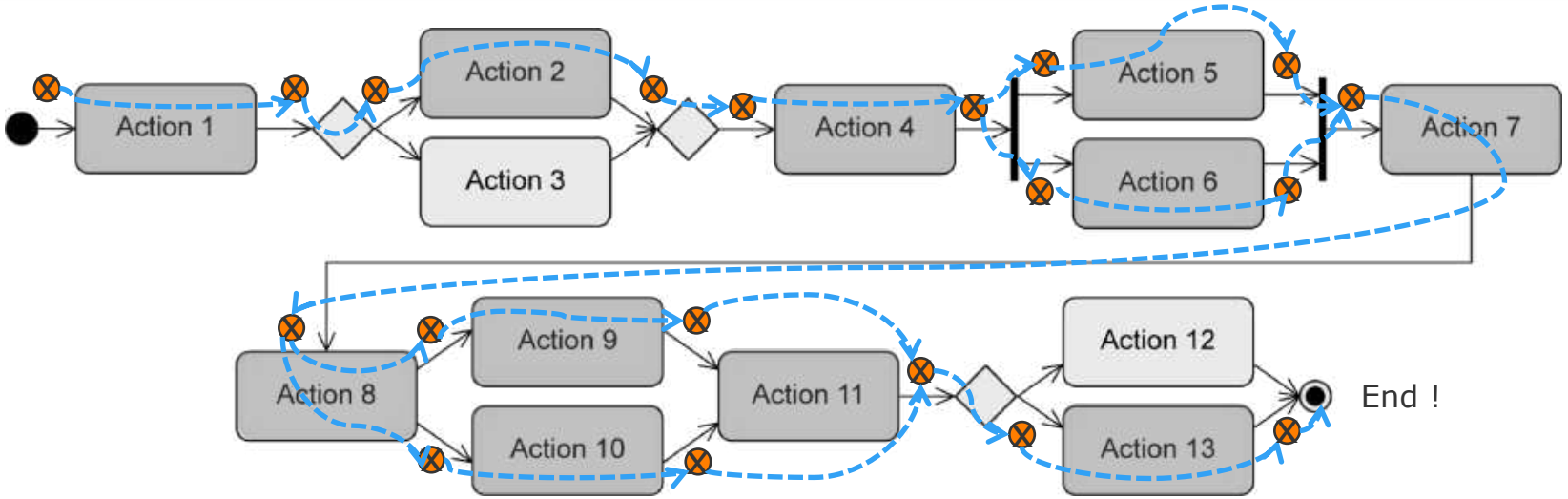
Contoh: Mengadakan Kuliah (Perspektif Mahasiswa)



TIDAK sama … mengapa?



Contoh: Token (Control Flow)



… semua *outgoing edge* dari *initial node* diberikan sebuah token….

… jika semua *incoming edge* dari sebuah action memiliki sebuah token, action diaktifkan dan siap dieksekusi

… sebelum eksekusi, action mengkonsumsi sebuah token dari setiap *incoming edge*;
setelah eksekusi, action meneruskan sebuah token ke setiap *outgoing edge*

… sebuah *decision node* memberikan token ke **satu** *outgoing edge* (ditentukan oleh nilai guard)

… sebuah *merge node* memberikan setiap token yang didapatkan ke *outgoing edge*

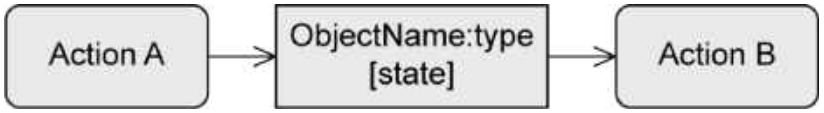
… sebuah *parallelization node* menggandakan token untuk **semua** *outgoing edge*

… sebuah *synchronization node* menunggu sampai semua incoming edges memiliki sebuah token,
menggabungkannya menjadi sebuah token dan memberikannya ke *outgoing edge*

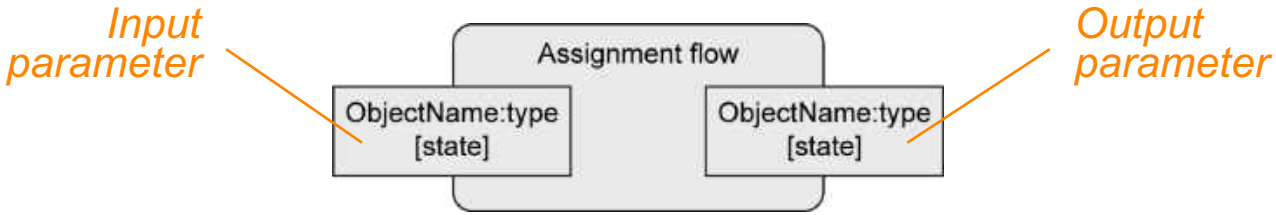
… token pertama yang mencapai *activity final node* menghentikan seluruh activity

Object Node

- Berisi object token
- Menggambarkan pertukaran data/object
- Merupakan sumber dan target sebuah object flow edge
- Information tambahan: tipe, state

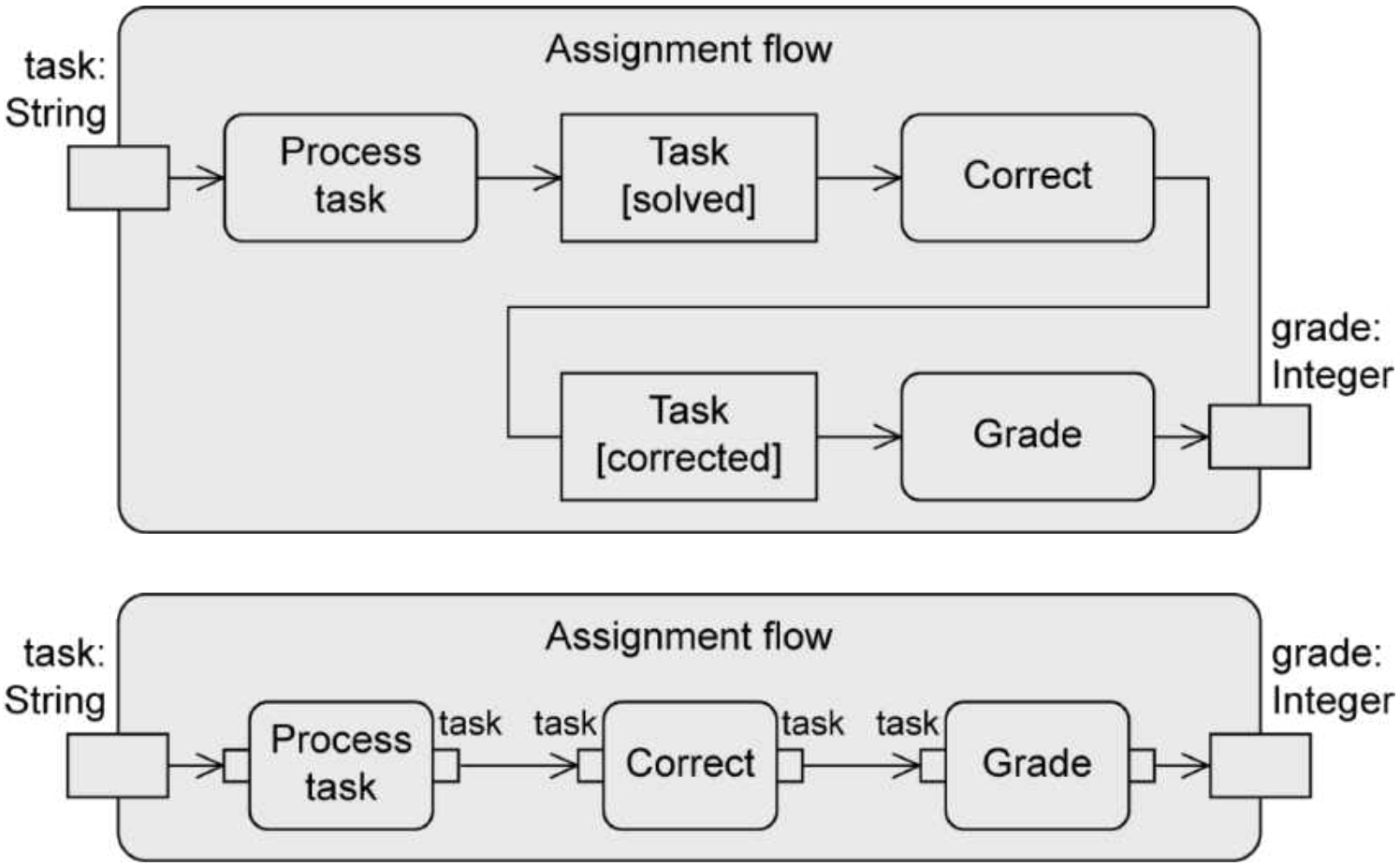


- Variasi notasi: object node sebagai parameter
  - Untuk activity



  - Untuk action (“pins”)

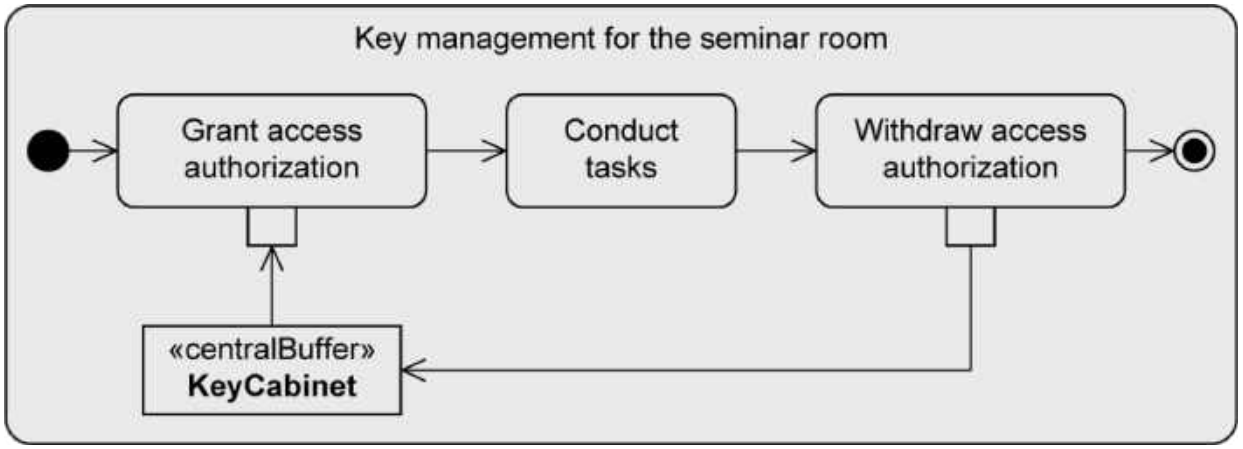
Contoh: Object Node



Central Buffer

«centralBuffer»
CB

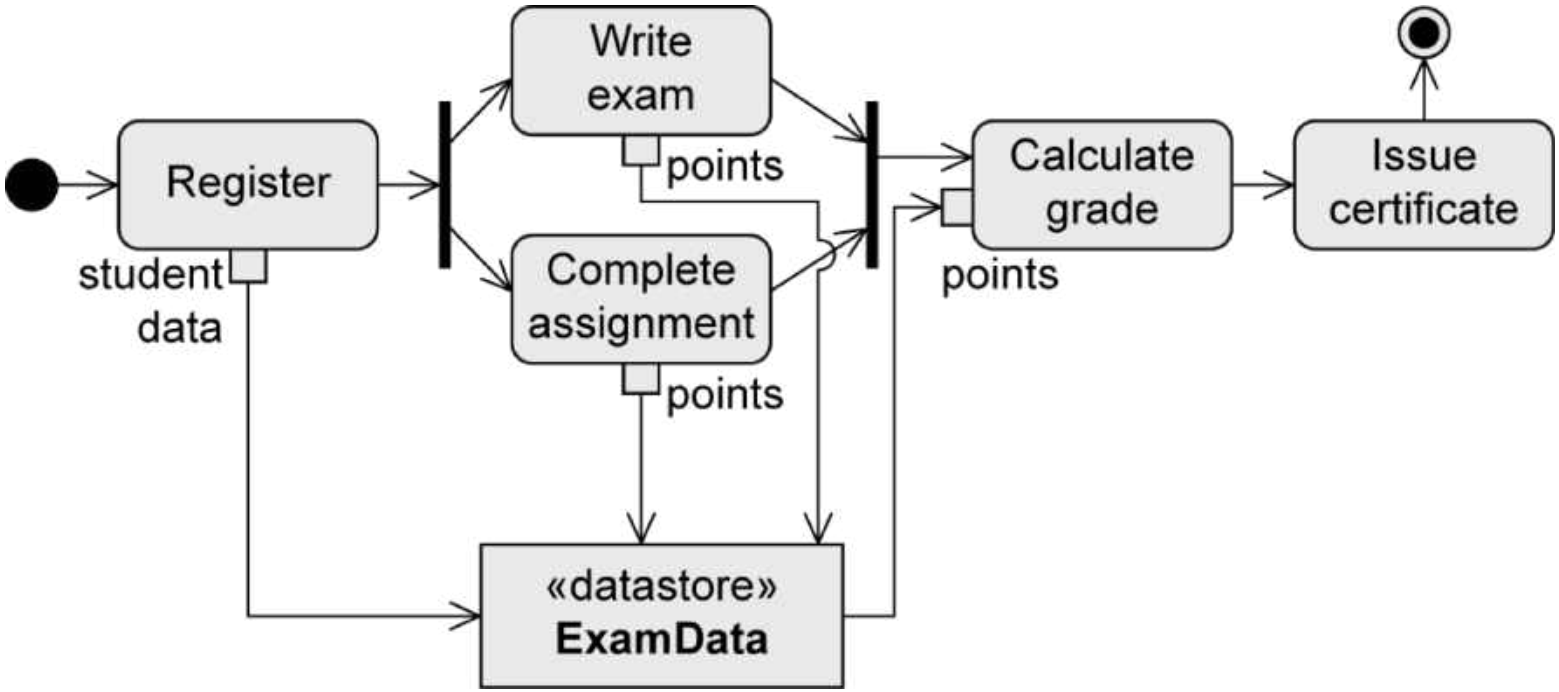
- Untuk menyimpan dan meneruskan object token
- Transient memory
- Menerima object token yang masuk dari object node dan meneruskannya ke object node lain
- Saat sebuah object token dibaca dari central buffer, token tersebut dihapus dari central buffer tidak dapat dikonsumsi



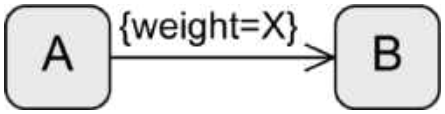
Data Store

«datastore»
DS

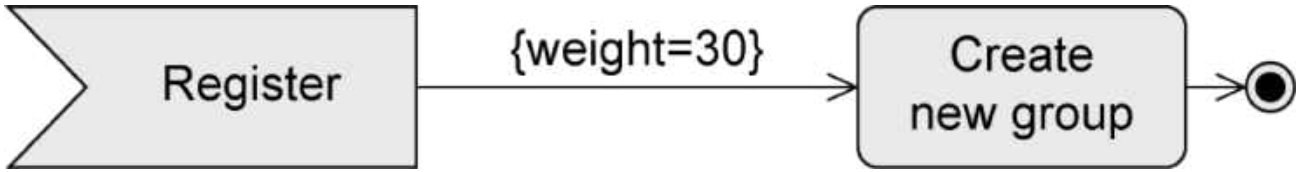
- Untuk menyimpan dan meneruskan object token
- Permanent memory
- Menyimpan object token secara permanen, meneruskan tiruan ke node lain



Berat Sebuah Edge



- Minimal jumlah token yang harus ada agar sebuah action dapat dieksekusi
- Default: **`1`**
- Semua token yang ada harus dikonsumsi: **`0`** (atau **`all`** atau **`*`**)



Connector

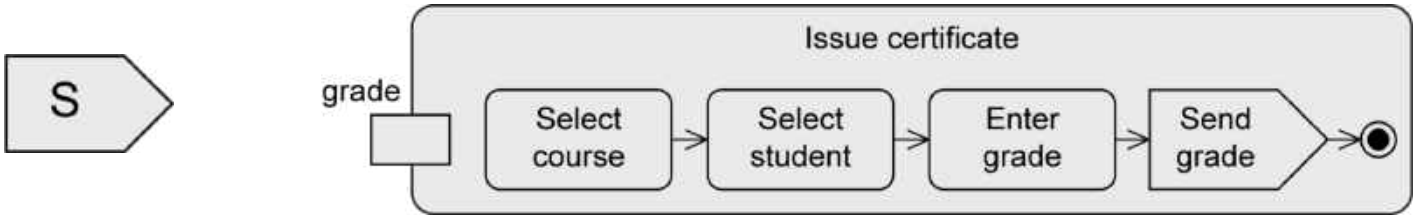
- Digunakan jika dua action yang berurutan saling berjauhan dalam diagram

- Tanpa connector:

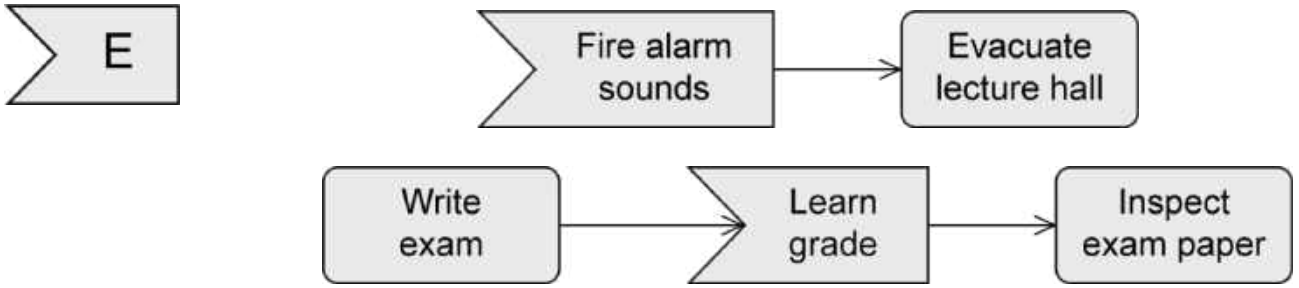
- Dengan connector

Event-Based Actions

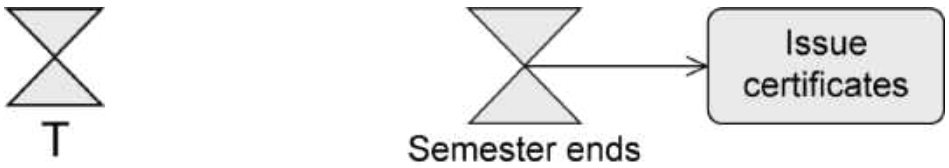
- Untuk mengirim signal
  - Mnegirim signal action



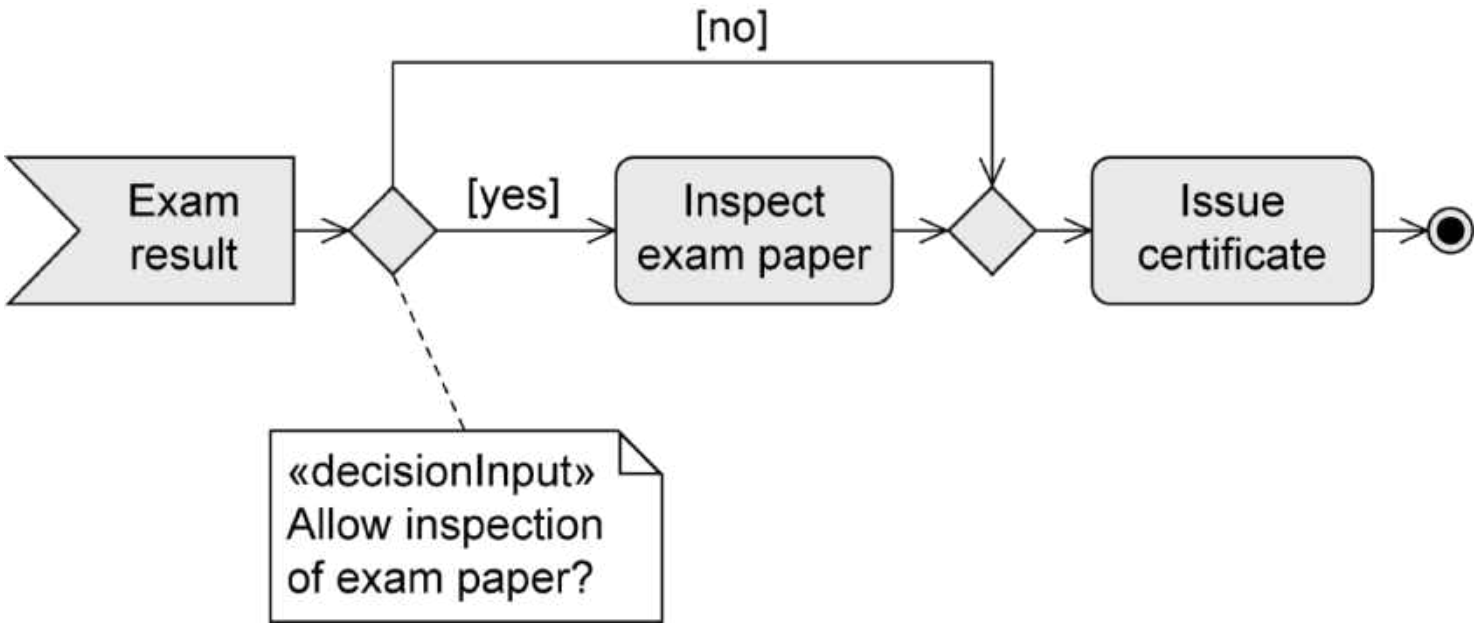
- Untuk menerima events
  - Menerima event action



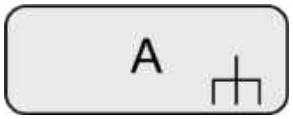
  - Menerima time event action



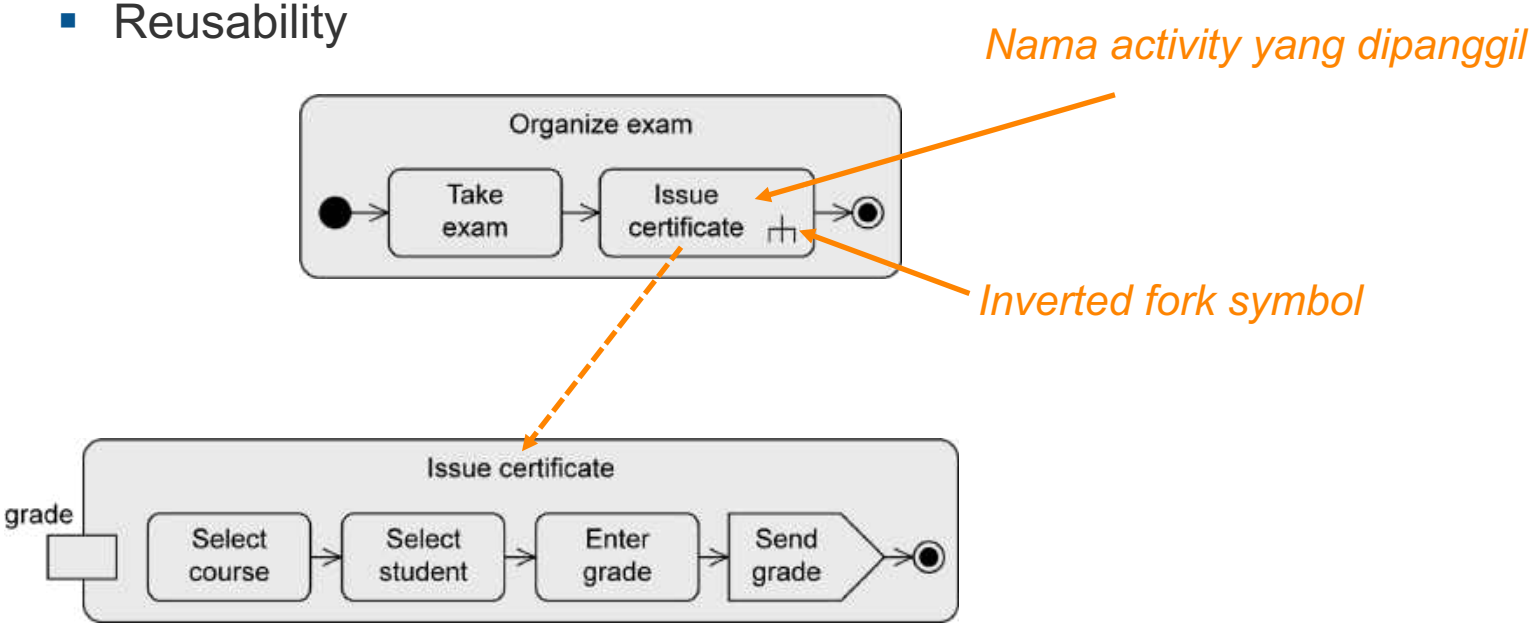
Contoh: Menerima Event Action



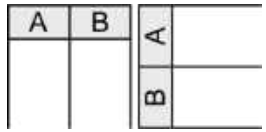
Call Behavior Action



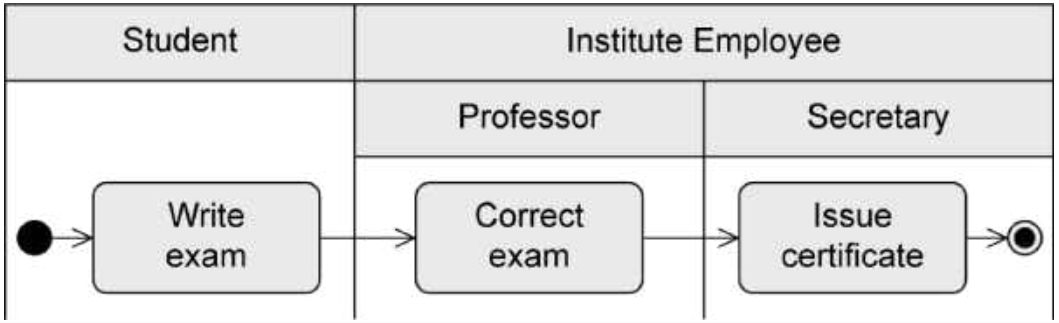
- Eksekusi sebuah action dapat memanggil sebuah activity
- Isi dari activity yang dipanggil dapat digambarkan di tempat lain
- Keuntungan:
  - Model menjadi lebih jelas
  - Reusability



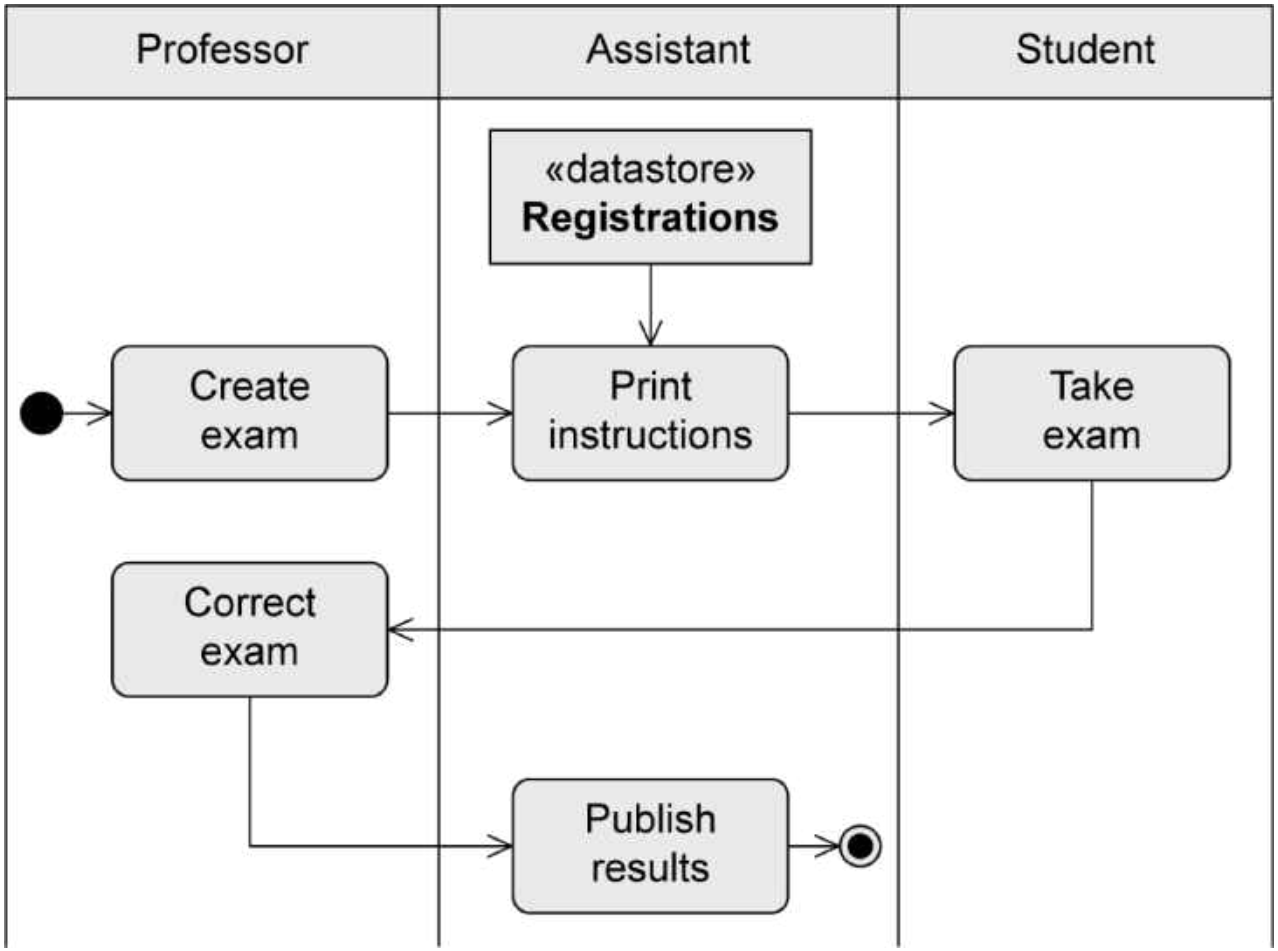
Partition



- “Swimlane”
- Secara grafis atau tekstual
- Memungkinkan pengelompokan node dan edge sebuah activity terkait tanggung jawab
- Tanggung jawab mencerminkan struktur organisasi atau roles
- Membuat diagram lebih tersetruktur
- Tidak mengubah semantic atau arti ekseskusi
- Contoh: partition **`Student`** dan **`Institute Employee`** (dengan subpartitions **`Professor`** dan **`Secretary`**)



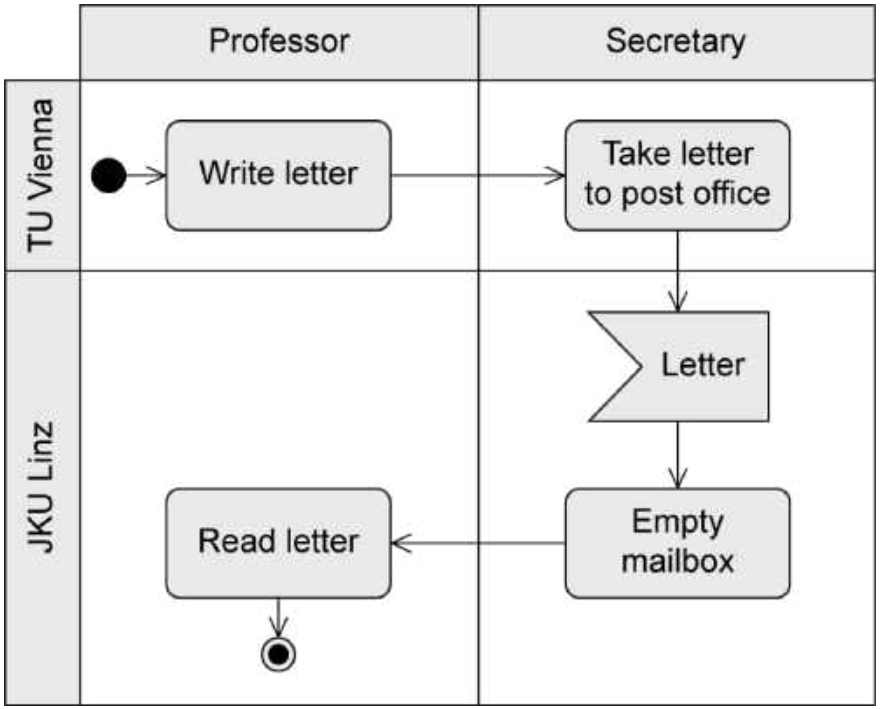
Contoh: Partitions

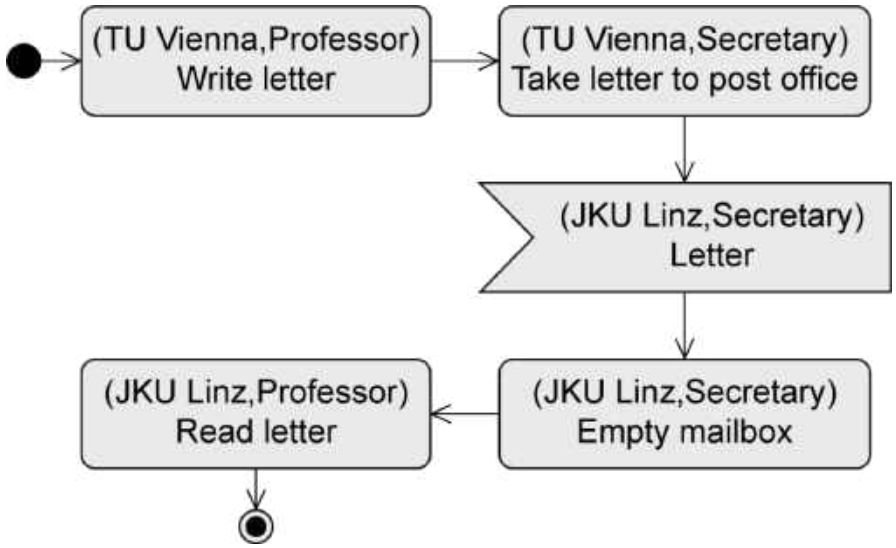


Multidimensional Partitions

- Notasi grafis

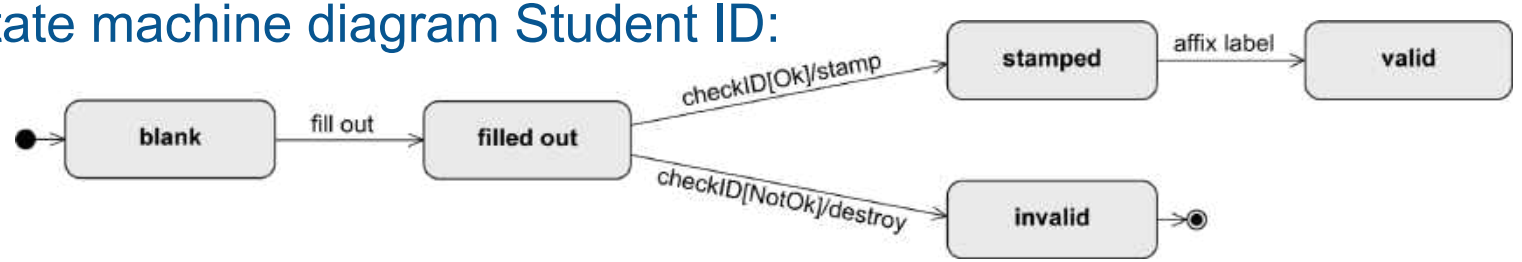
… atau notasi tekstual



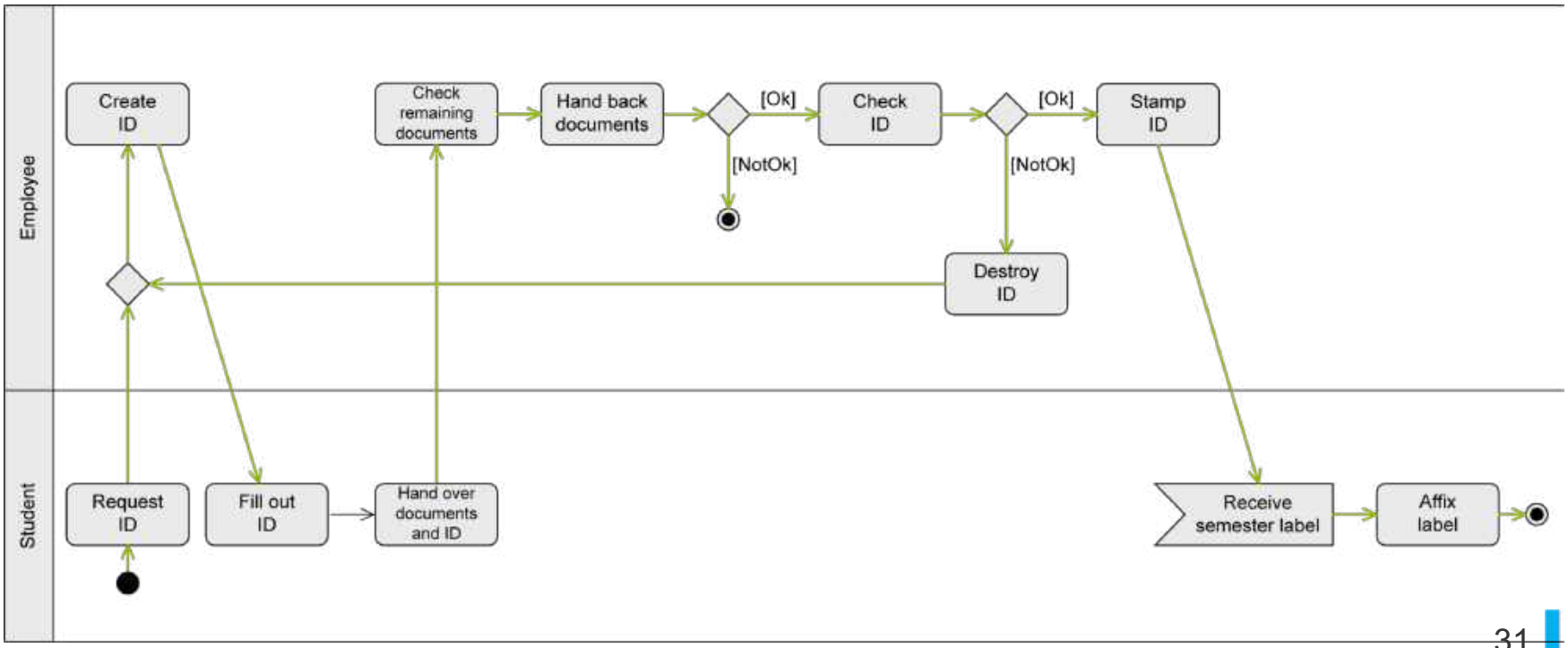


Contoh: Menerbitkan Student ID (1/2)

- State machine diagram Student ID:

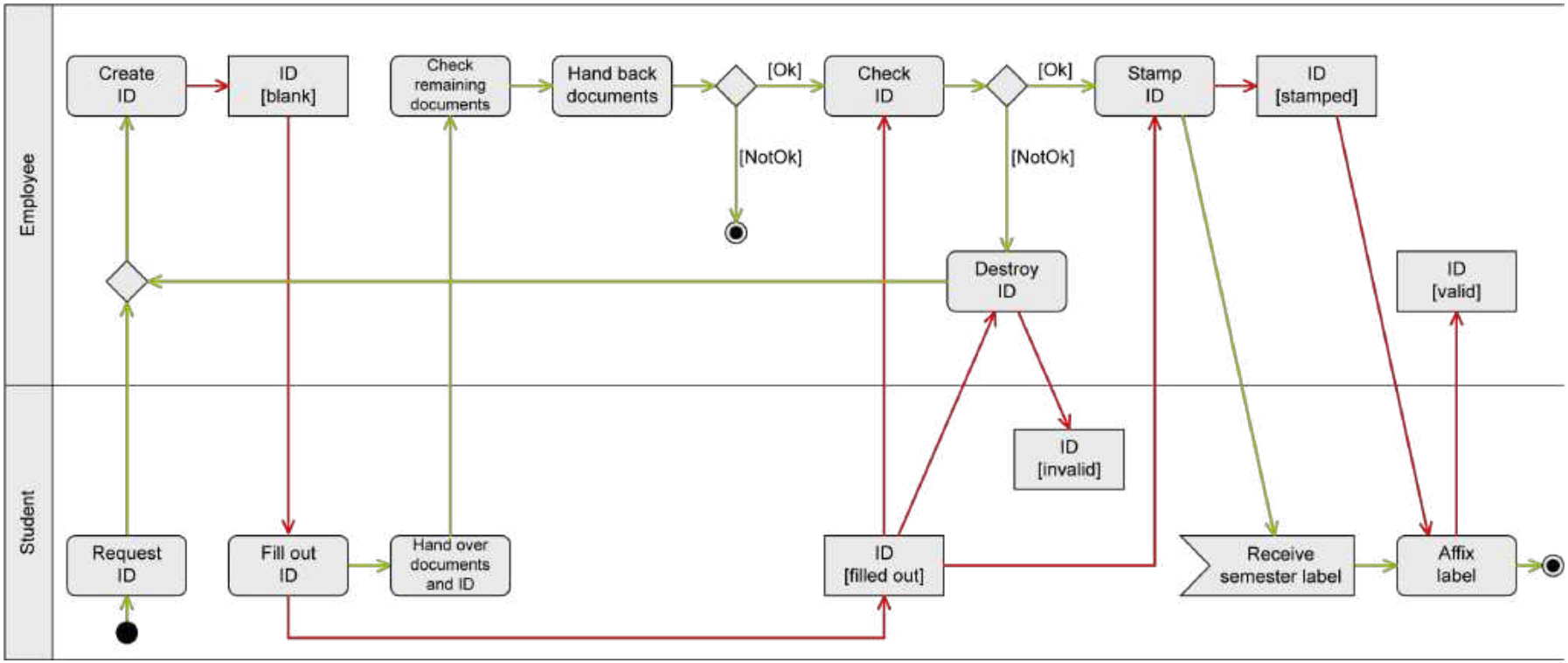


- Activity diagram – control flow:



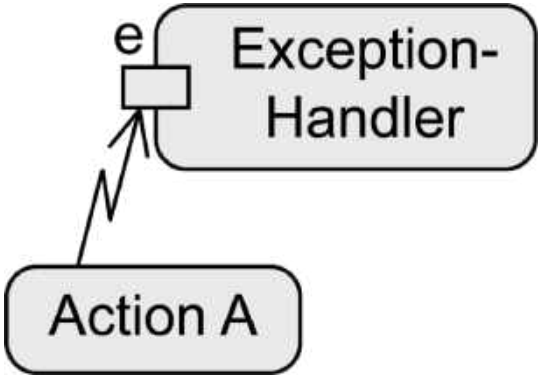
Example: Menerbitkan Student ID (2/2)

- Control flow (green) dan object flow (red) dalam sebuah activity diagram



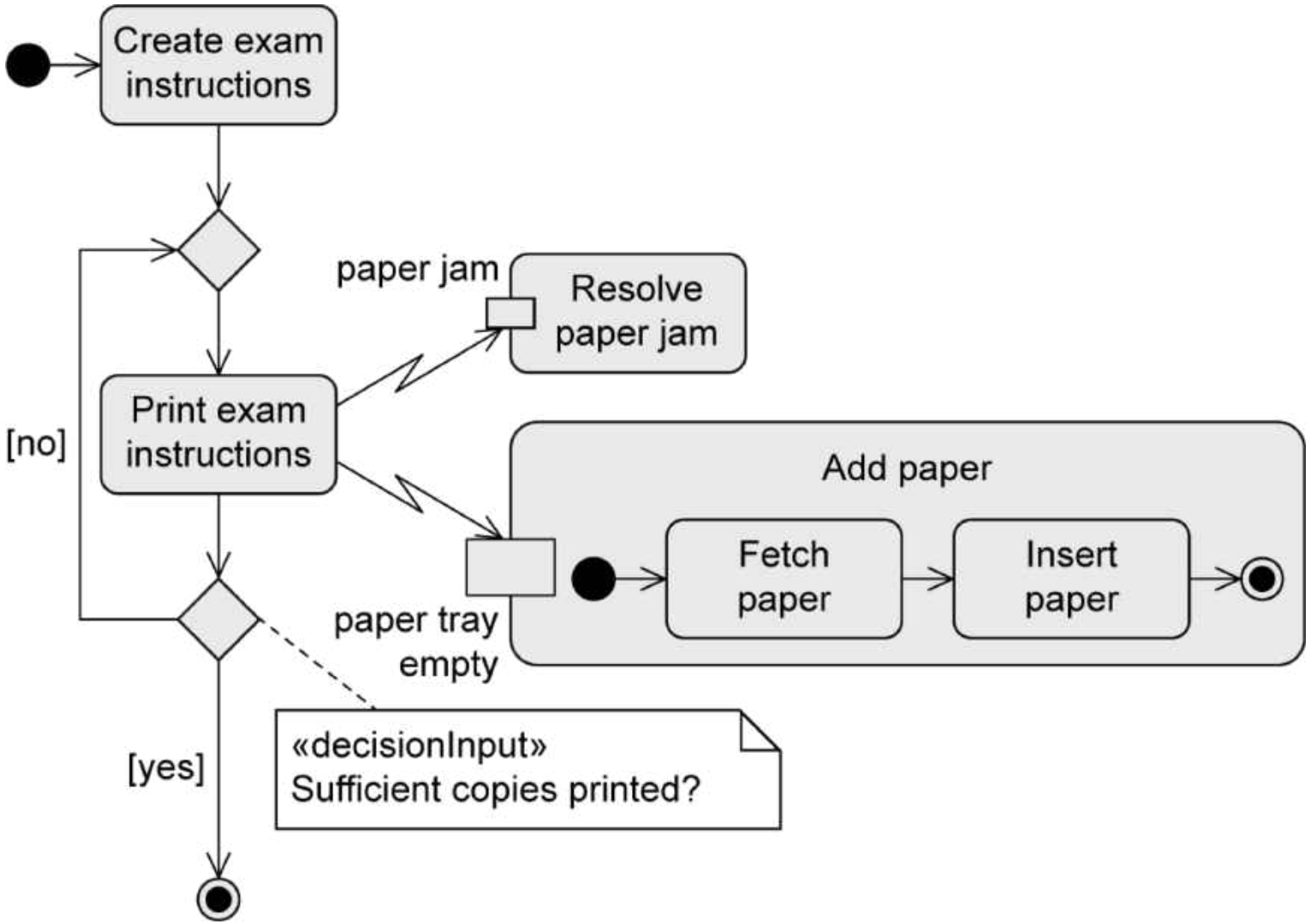
Exception Handling – Exception Handler

- Exceptions yang telah terdefinisi
- Menggambarkan bagaimana sistem harus bereaksi dalam situasi error tertentu
- Exception handler menggantikan action dimana error terjadi



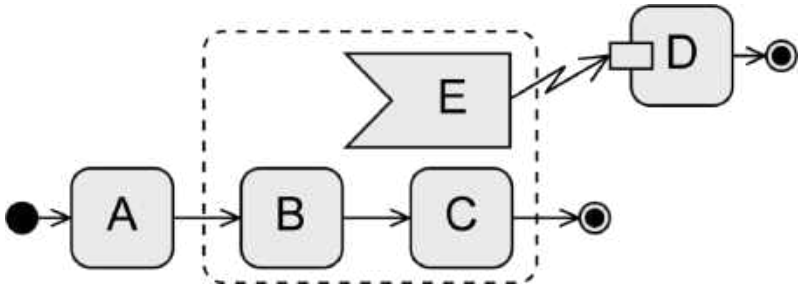
- Jika error **`e`** terjadi…
  - Semua token dalam **`Action A`** dihapus
  - Exception handler diaktifkan
  - Exception handler dieksekusi menggantikan **`Action A`**
  - Eksekusi berlajut seperti biasa

Contoh: Exception Handler



Exception Handling– Interruptible Activity Region

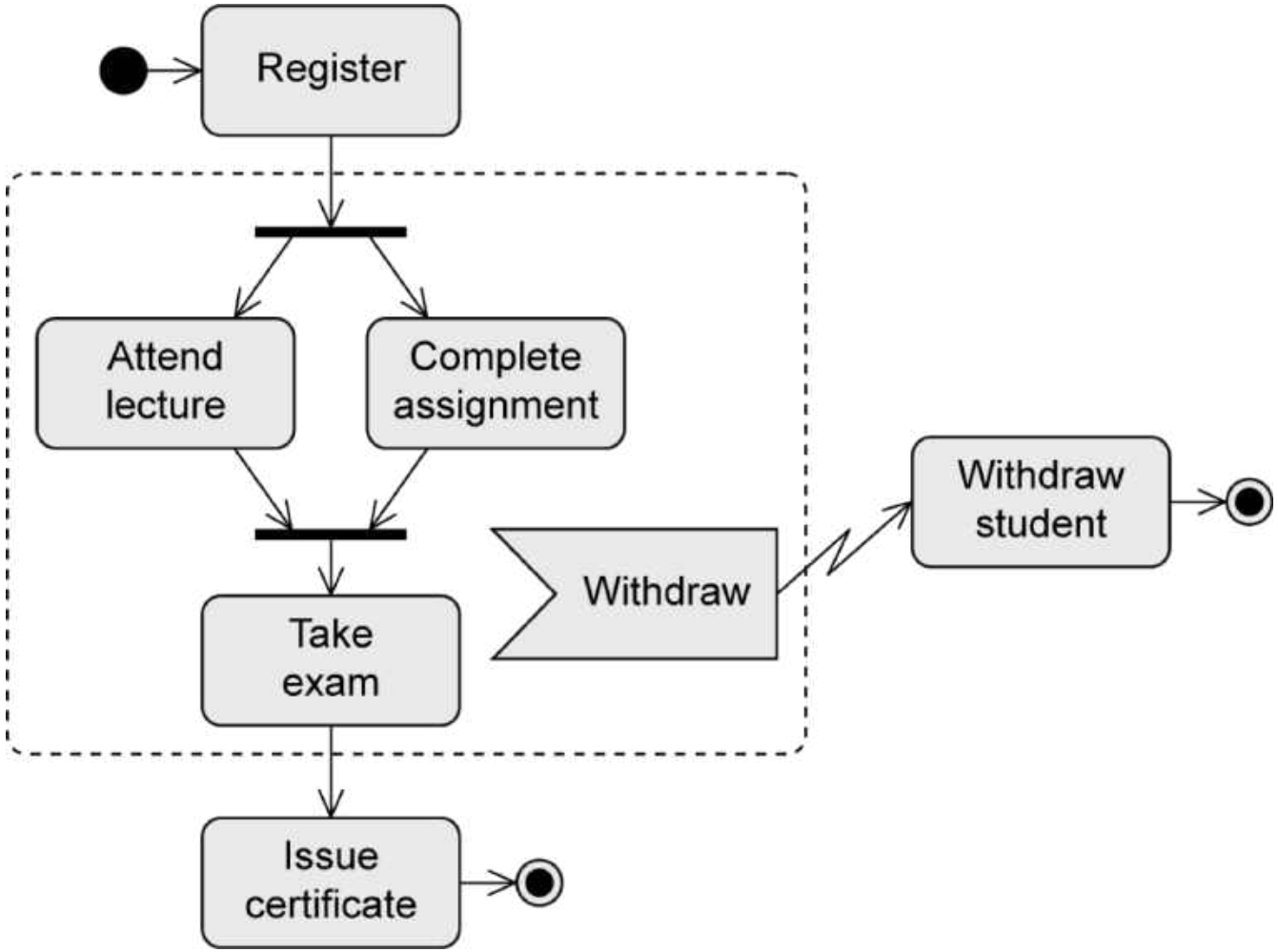
- Menggambarkan sekelompok action yang eksekusinya langsung dihentikan jika sebuah event tertentu terjadi occurs. Dalam kasus tersebut, perilaku yang lain yang dieksekusi.



- Jika **E** terjadi saat **B** atau **C** dieksekusi
  - Exception handling diaktifkan
  - Semua control token dalam kotak garis putus-putus (= dalam **B** dan **C**) dihapus
  - **D** diaktifkan dan dieksekusi

- Tidak ada “kembali” eksekusi seperti biasa!

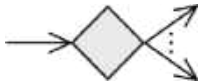
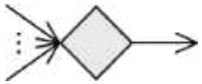
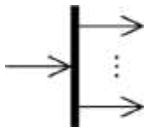
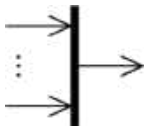
Contoh: Interruptible Activity Region



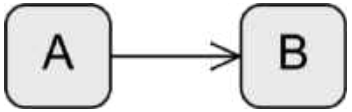
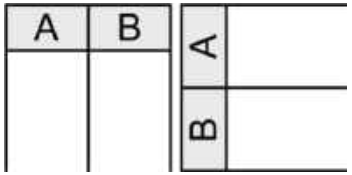
Notasi Elemen (1/5)

Name	Notation	Description
Action node		Menggambarkan sebuah aksi (atomik!)
Activity node		Menggambarkan sebuah an activity (dapat dipecah lagi)
Initial node		Awal eksekusi sebuah activity
Activity final node		Akhir SEMUA execution paths sebuah activity

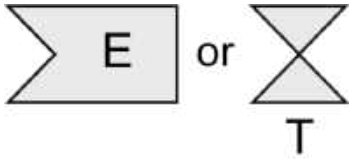
Notation Elements (2/5)

Name	Notation	Description
Decision node		Memisahkan satu execution path menjadi alternative execution path
Merge node		Menggabungkan alternative execution path menjadi satujalur eksekusi
Parallelization node		Memisahkan satu execution path menjadi concurrent execution path
Synchronization node		Menggabungkan concurrent execution path mejadi satu execution path

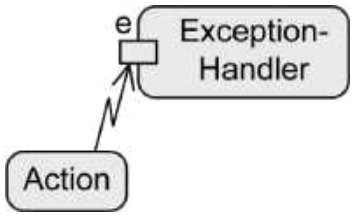
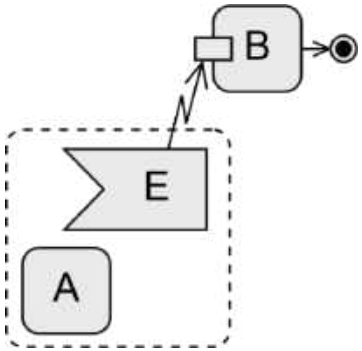
Notation Elements (3/5)

Name	Notation	Description
Flow final node		Akhir SEBUAH jalur eksekusi activity
Edge		Hubungan antar nodes dalam sebuah activity
Call behavior action		Action A merujuk pada sebuah activity dengan nama yang sama
Partition		Mengelompokkan node dan edge dalam sebuah activity

Notasi Elemen (4/5)

Name	Notation	Description
Send signal action		Pengiriman signal ke penerima
Asynchronous accept (timing) event action		Menunggu event `E` atau time event `T`
Object node		Mengandung data atau object
Parameter for activities Parameter for actions (pins)	 	Mengandung data dan object sebagai parameter input dan output

Notasi Elemen (5/5)

Name	Notation	Description
Exception Handler		Exception handler dijalankan jika terjadi **e**
Interruptible activity region		Aliran berlanjut pada jalur yang berbeda jika event **E** terdeteksi