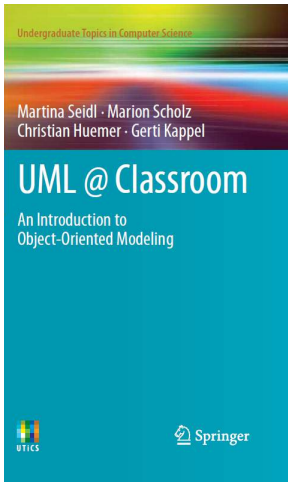
Object-Oriented Modeling

Structure Modeling

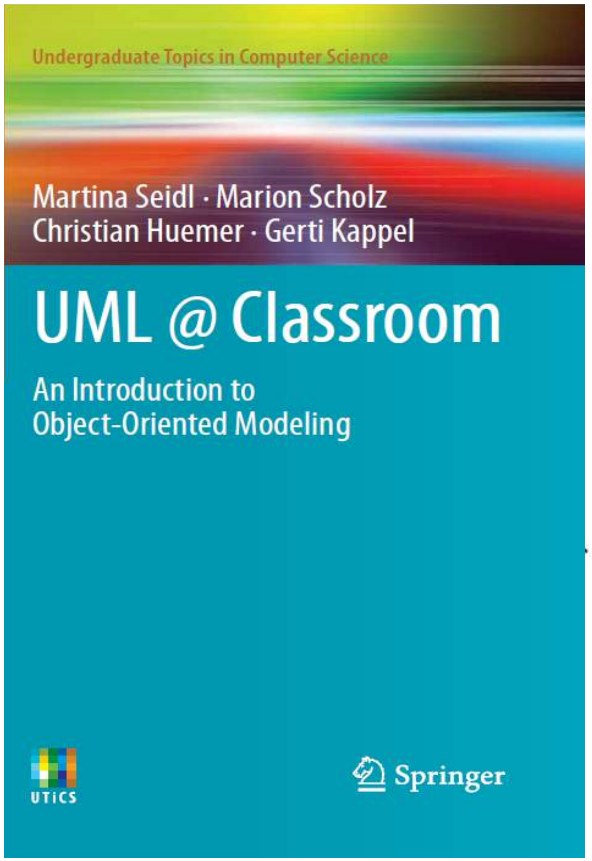


Slide untuk melengkapi buku UML@Classroom
Versi 1.0
Diterjemahkan dari
slide milik Busines Informatics Group, Vienna University of Technology

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

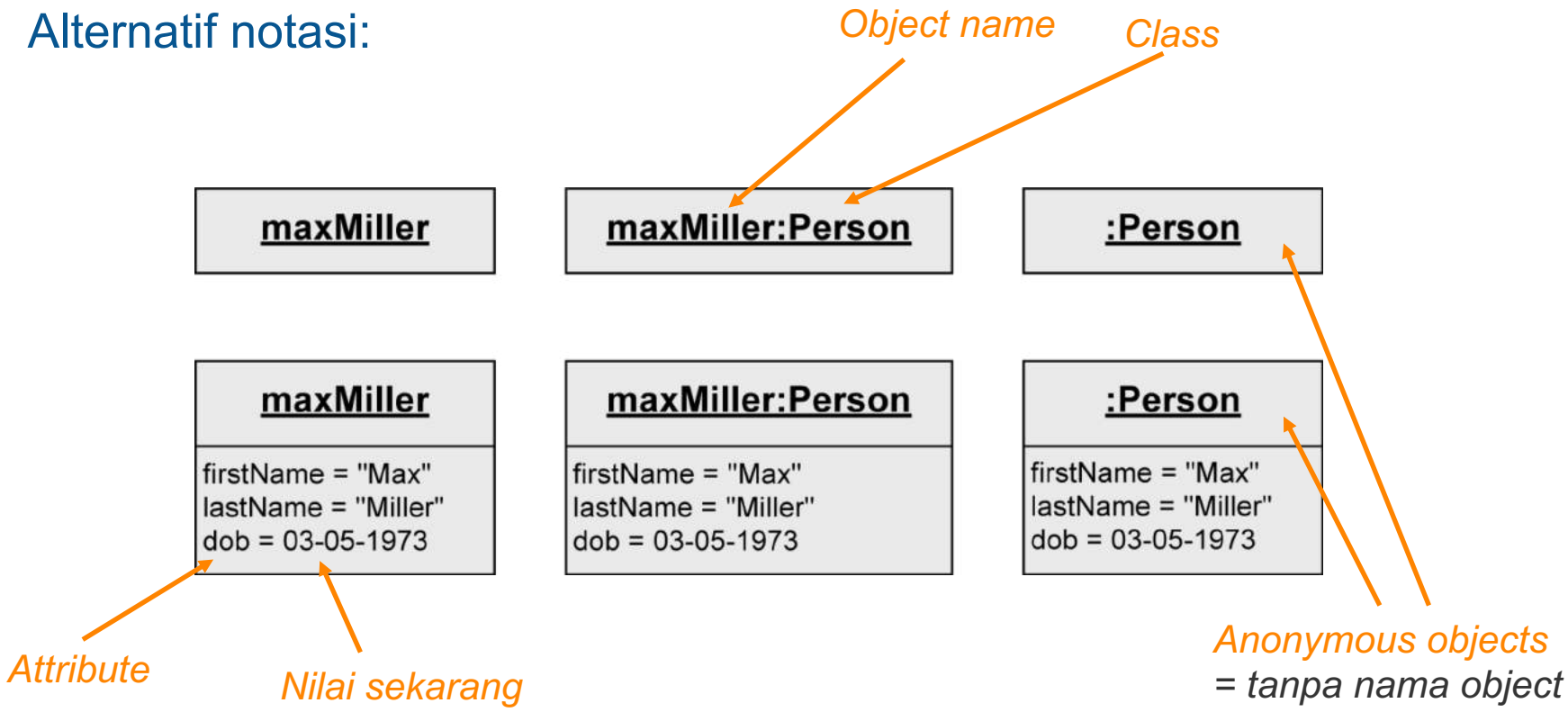
- Use Case Diagram
- **Structure Modeling**
- State Machine Diagram
- Sequence Diagram
- Activity Diagram

Materi

- Objects
- Classes
- Attributes
- Operations
- Relationships
  - Binary Association
  - N-ary Association
  - Association Class
  - Aggregation
  - Generalization
- Membangun class diagram
- Code Generation

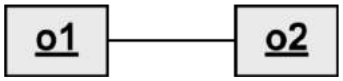
Object

- Individual dari sebuah sistem
- Alternatif notasi:

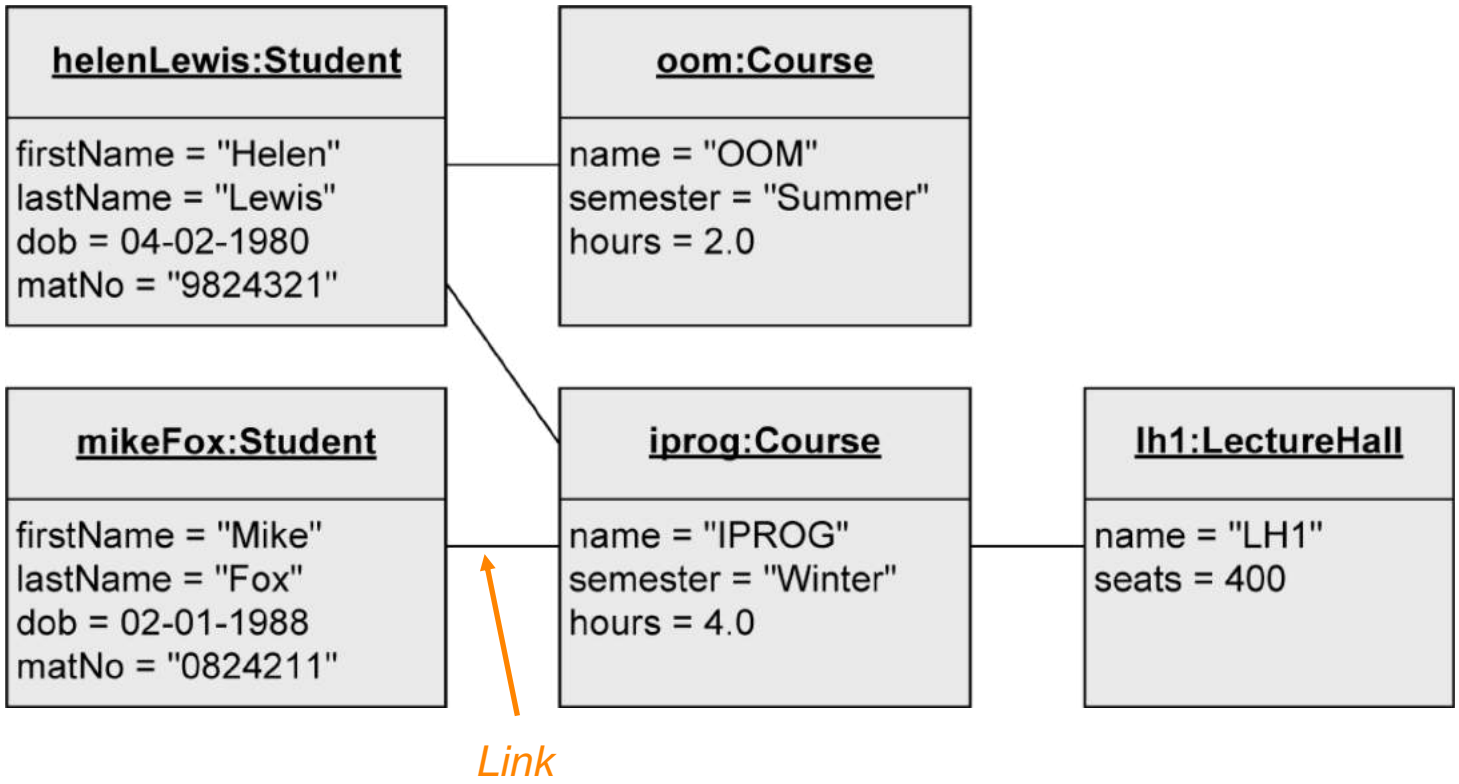


© Polibatam

Object Diagram



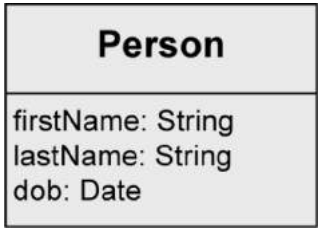
- Object dari sebuah sistem dan relationship mereka (link)
- Gambaran berbagai object pada suatu saat



Dari Object ke Class

- Individual dari sebuah sistem seringkali memiliki karakteristik dan perilaku yang sama
- Sebuah class adalah sebuah rencana konstruksi untuk sekumpulan object yang sama dalam sebuah sistem

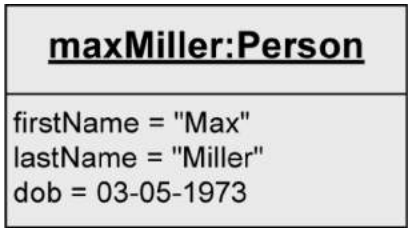
Class



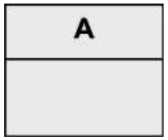
- Object merupakan instance dari class
- Attributes: karakteristik stuktural sebuah class
  - Nilai yang berbeda untuk setiap instance (= object)

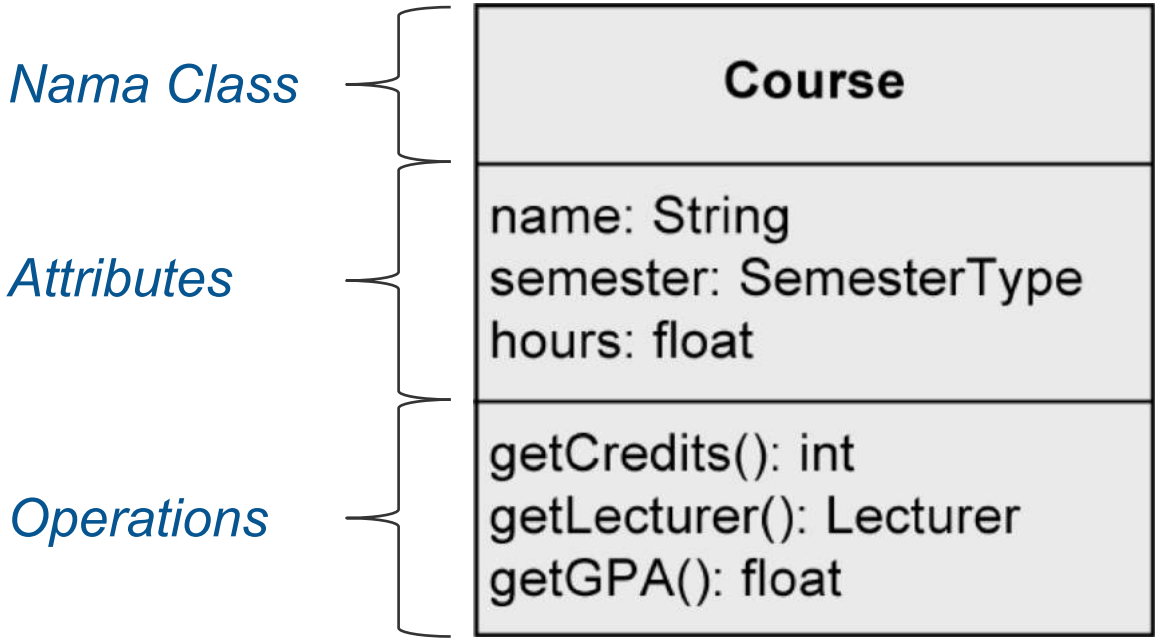
Object dari class tersebut

- Operations: perilaku dari class
  - Identik untuk semua object dari sebuah class
    → tidak digambarkan dalam object diagram



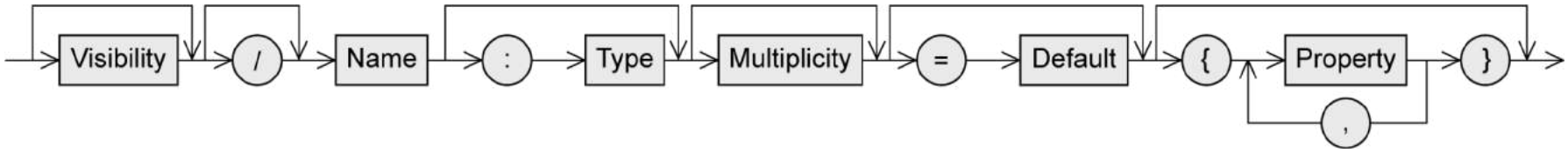
Class



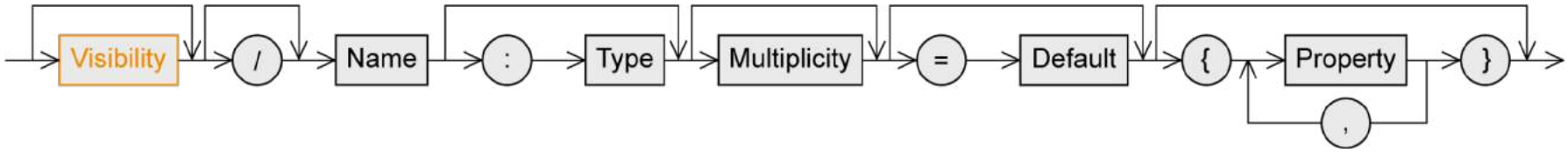


© Polibatam

Syntax Attribute



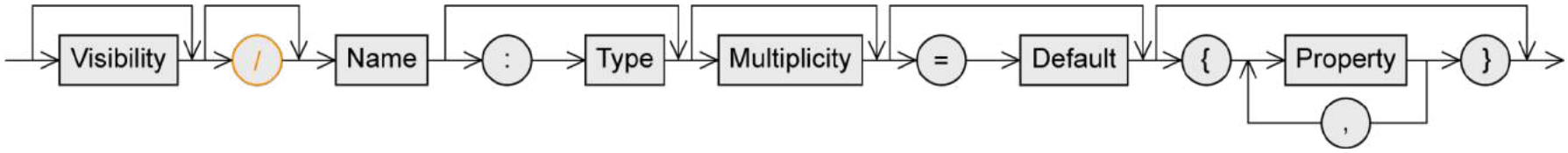
Syntax Attribute - Visibility



Person
+ firstName: String + lastName: String – dob: Date # address: String[1..*] {unique, ordered} – ssNo: String {readOnly} – /age: int – password: String = "pw123" – personsNumber: int

- Siapa yang diperbolehkan mengakses attribute
  - + ... public: semua orang
  - - ... private: hanya object itu sendiri
  - # ... protected: class itu sendiri dan subclass-nya
  - ~ ... package: semua class dalam package yang sama

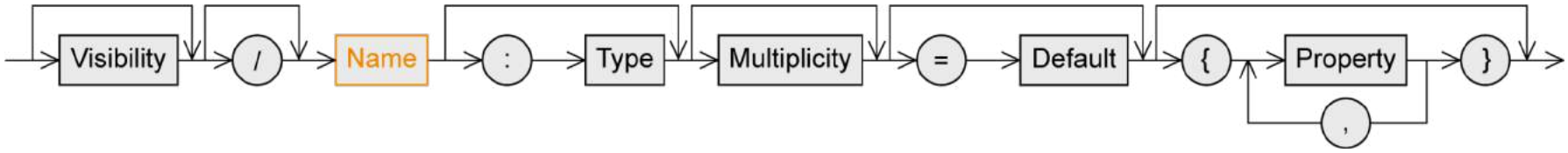
Syntax Attribute - Derived Attribute



Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" personsNumber: int

- Nilai attribute diturunkan dari attribute lain
  - **age**: dihitung dari *date of birth*

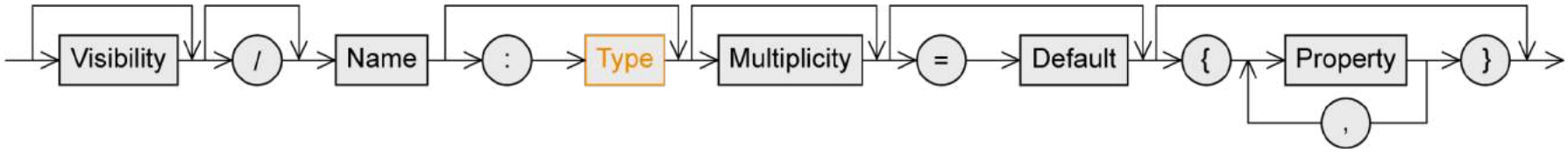
Syntax Attribute - Name



Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" personsNumber: int

- Nama attribute

Syntax Attribute - Type



Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" personsNumber: int

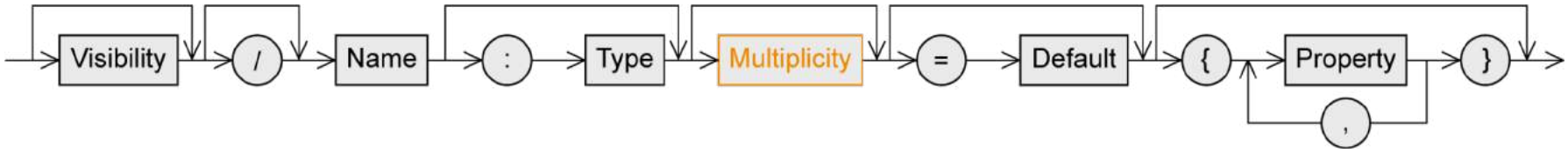
- Type
  - User-defined class
  - Tipe data
    - Tipa data primitif
      - Pre-defined: Boolean, Integer, UnlimitedNatural, String
      - User-defined: «**primitive**»
      - Composite data type: «**datatype**»
    - Enumerations: «**enumeration**»

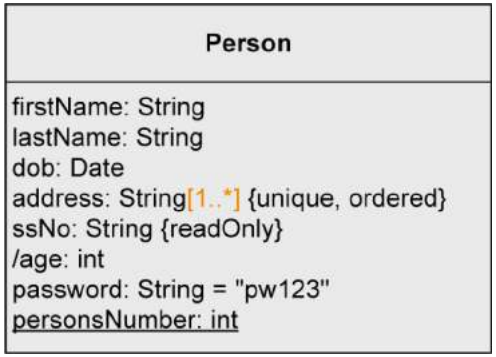
«primitive» **Float**
round(): void

«datatype» **Date**
day month year

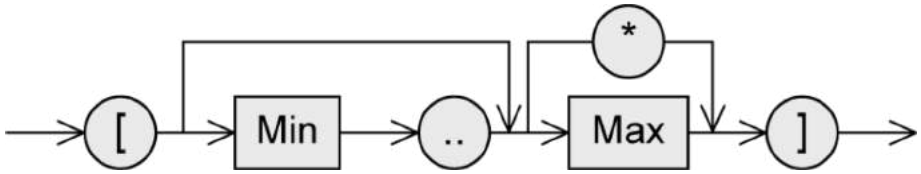
«enumeration» **AcademicDegree**
bachelor master phd

Syntax Attribute - Multiplicity



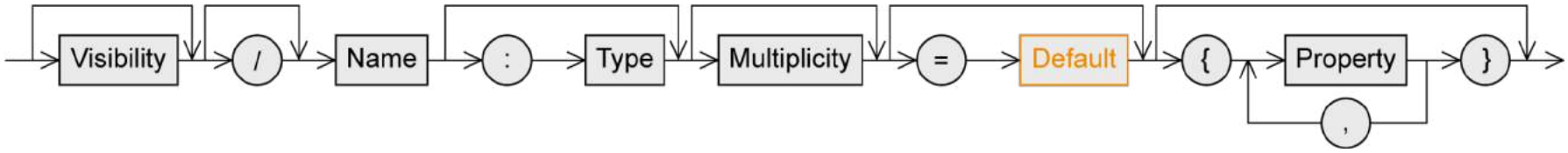


- Jumlah nilai yang dapat dimiliki oleh sebuah attribute
- Nilai default: 1
- Notasi: **[min..max]**
  - tanpa batas atas: `[*]` or `[0..*]`



© Polibatam

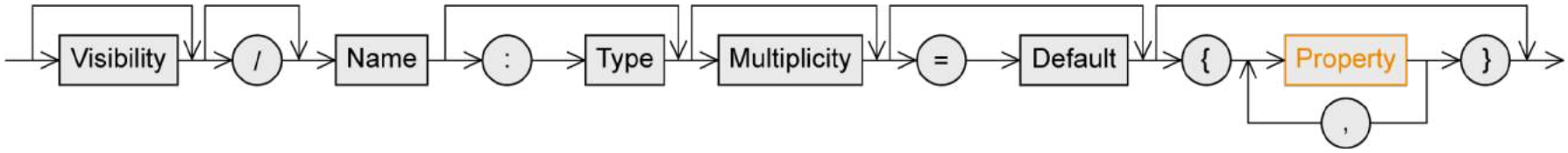
Syntax Attribute – Nilai Default

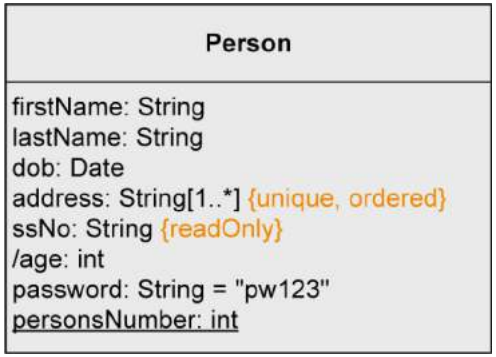


Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" personsNumber: int

- Nilai default
  - Digunakan jika nilai atribut tidak diisi secara eksplisit oleh user

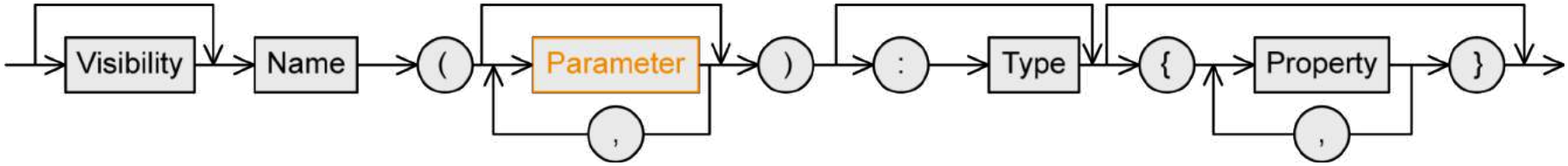
Syntax Attribute – Property

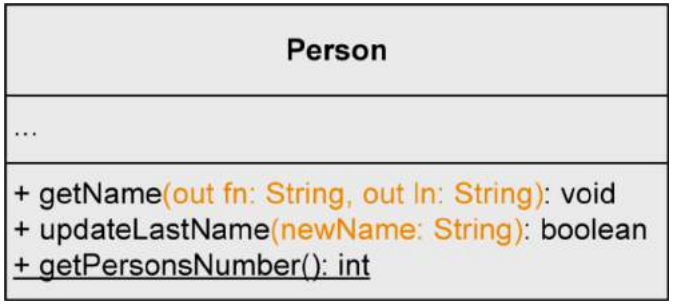




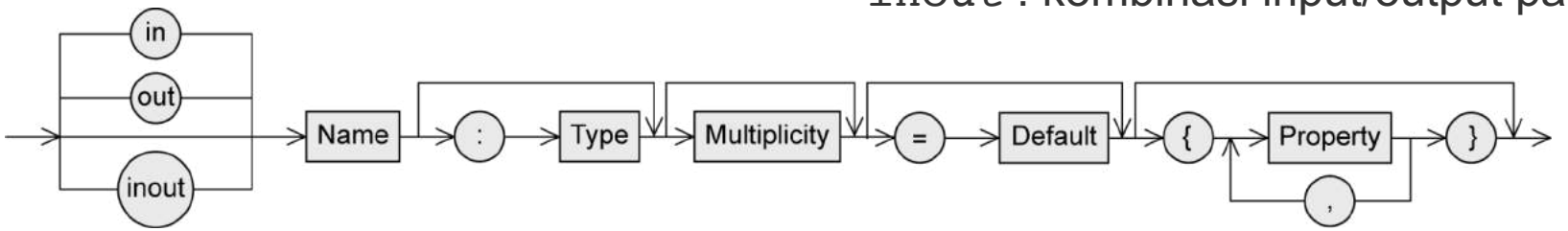
- Pre-defined property
  - `{readOnly}` … nilai tidak dapat diubah
  - `{unique}` … tidak boleh duplikasi
  - `{non-unique}` … boleh duplikasi
  - `{ordered}` … nilai terurut sesuai aturan tertentu
  - `{unordered}` … tidak ada ketentuan urutan nilai
- Spesifikasi attribute
  - Set: `{unordered, unique}`
  - Multi-set: `{unordered, non-unique}`
  - Ordered set: `{ordered, unique}`
  - List: `{ordered, non-unique}`

Syntax Operation - Parameters

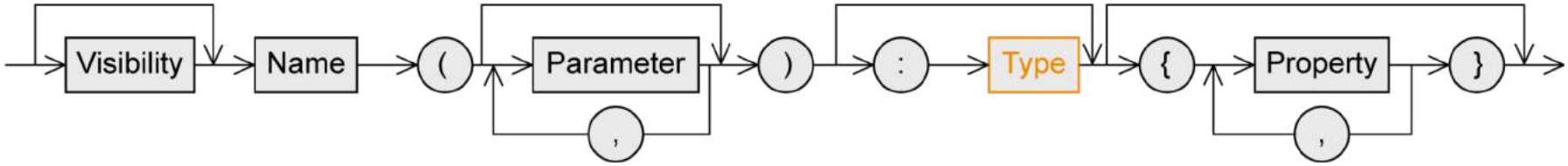




- Notasi mirip dengan attribute
- Arah parameter
  - `in` … input parameter
    - Ketika *operation* digunakan, sebuah nilai diharapkan dari parameter tersebut
  - `out` … output parameter
    - Setelah *operation* dijalankan, parameter mendapat nilai baru
  - `inout` : kombinasi input/output parameter



Syntax Operation - Type

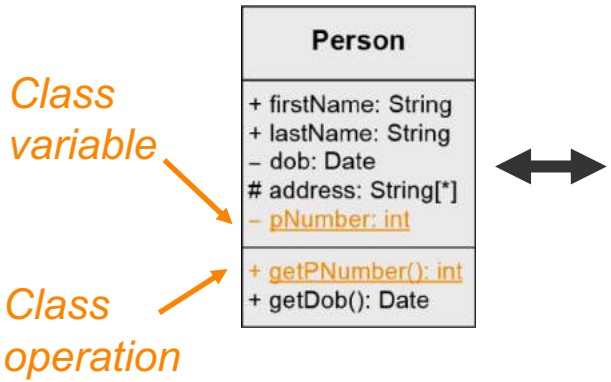


Person
...
getName(out fn: String, out ln: String): void updateLastName(newName: String): boolean getPersonsNumber(): int

- Type dari return value

Class Variable dan Class Operation

- Instance variable (= instance attribute): attribute didefinisikan pada level instance
- Class variable (= class attribute, static attribute)
  - Didefinisikan sekali per class, digunakan oleh semua instances dari class
  - Misal counter untuk menghitung jumlah instance sebuah class, constants, etc.
- Class operation (= static operation)
  - Dapat digunakan jika tidak ada instance dari class tersebut yang dibuat
  - Misal constructors, counting operations, math. functions (sin(x)), etc.
- Notasi: nama class variable / class operation bergaris bawah

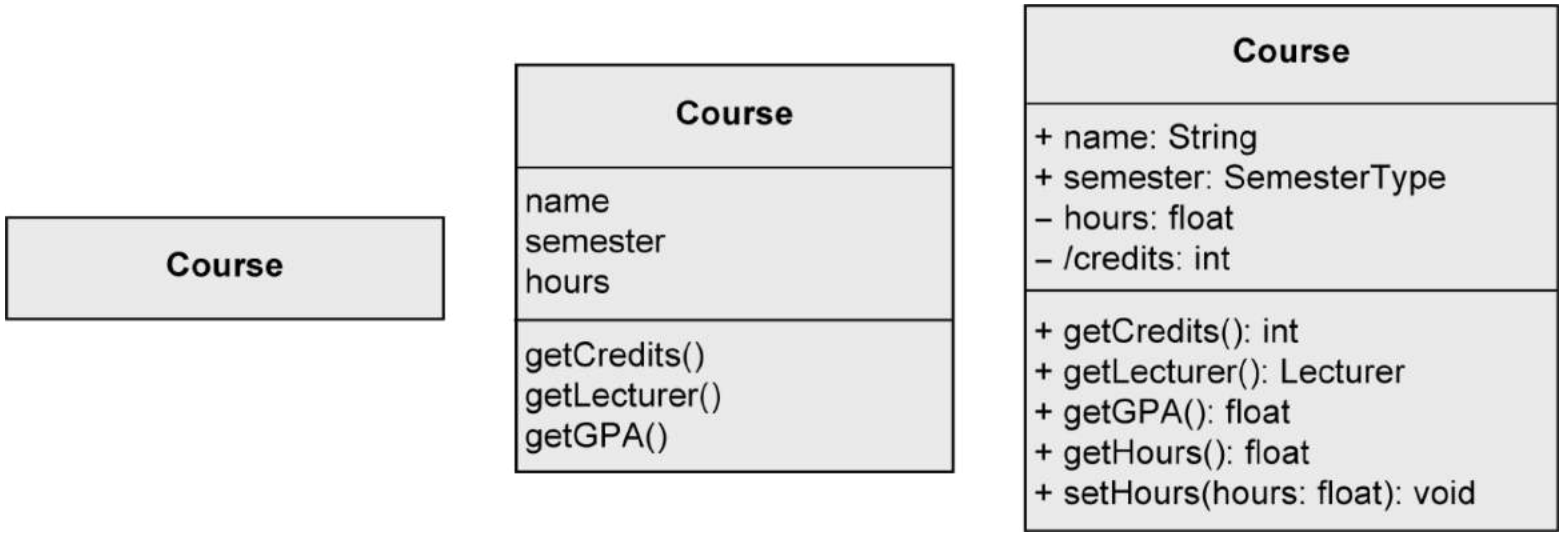



```
class Person {

   public String firstName;
   public String lastName;
   private Date dob;
   protected String[] address;
   private static int pNumber;
   public static int getPNumber() {…}
   public Date getDob() {…}
}
```

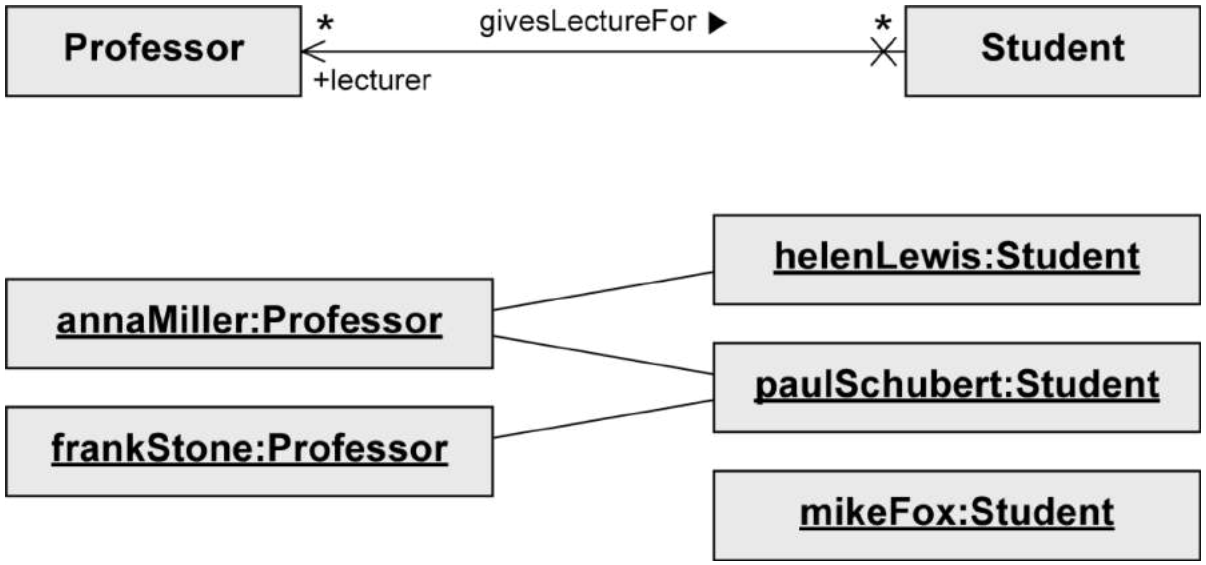
Spesifikasi Class: Berbagai Level Gambaran Detail Class

coarse-grained *fine-grained*

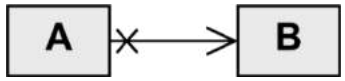


Association

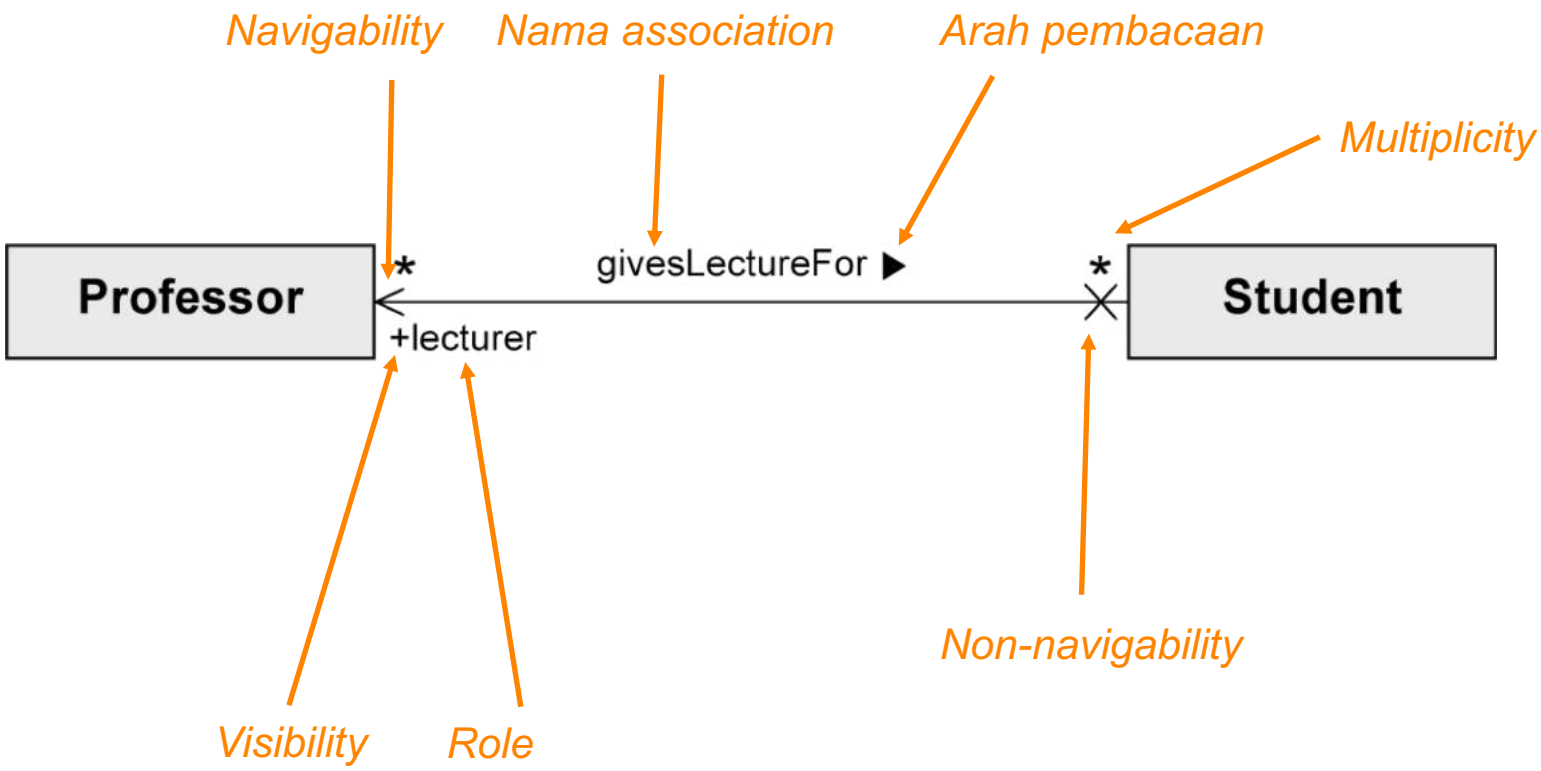
- Memodelkan relationship antar instance berbagai class



Binary Association



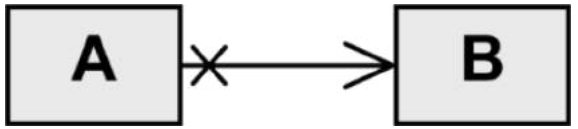
- Menghubungkan instance dari dua class

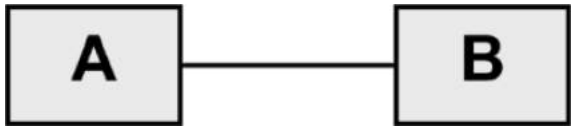


© Polibatam

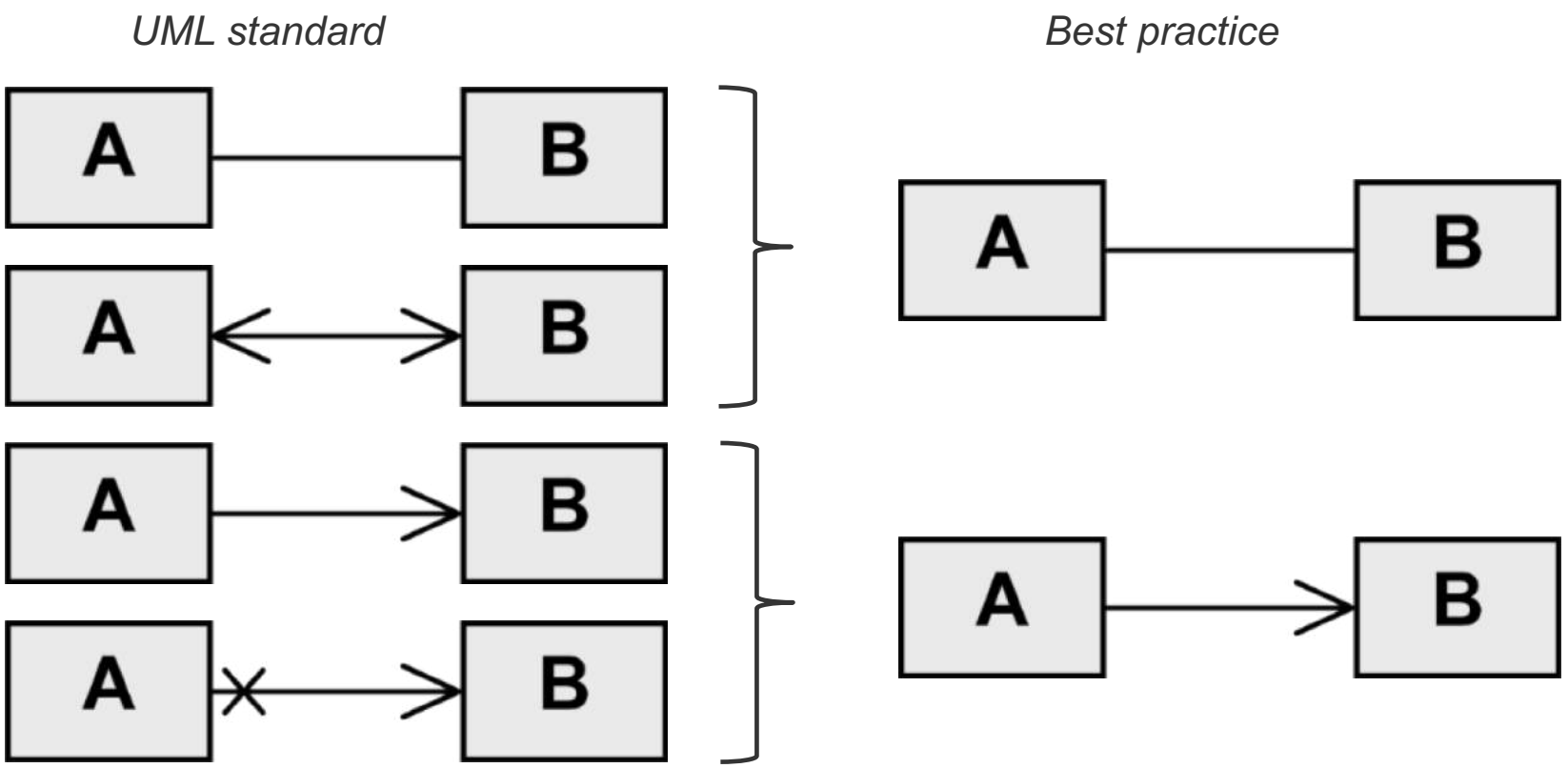
Binary Association - Navigability

- Navigability: sebuah object mengetahui object dari partnernya sehingga dapat mengakses attribute dan operation yang Nampak (*visible*)
  - Dilambangkan dengan ujung panah terbuka
- Non-navigability
  - Dilambangkan dengan tanda silang
- Contoh:
  - **A** dapat mengakses attribute dan operation milik **B** yang visible
  - **B** tidak dapat mengakses attribute dan operation apapun milik **A**
- Navigability tidak didefinisikan
  - Asumsi terdapat navigability ke kedua arah





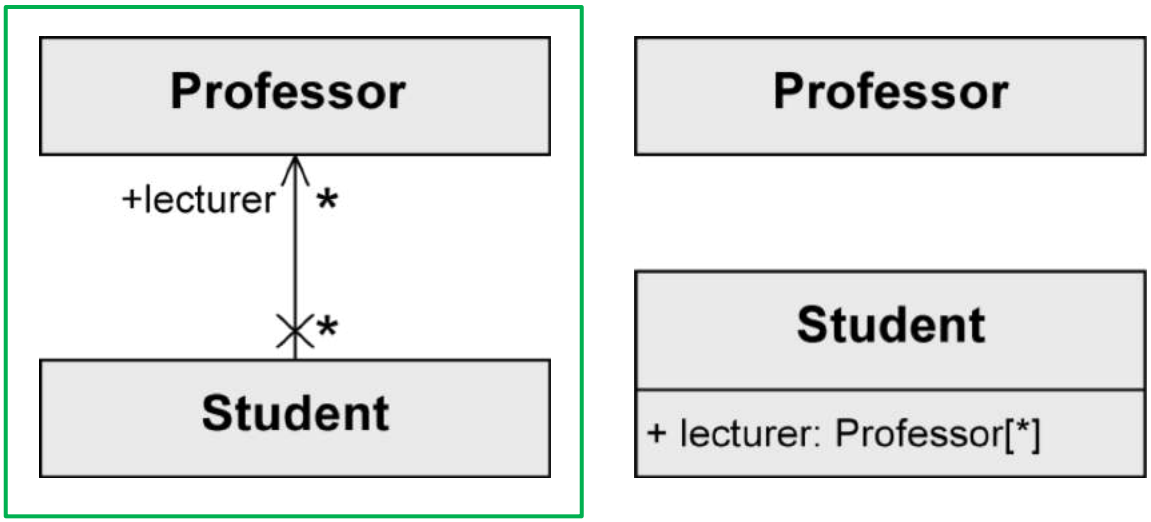
Navigability – UML Standard vs. Best Practice



© Polibatam

Binary Association sebagai Attribute

Lebih baik



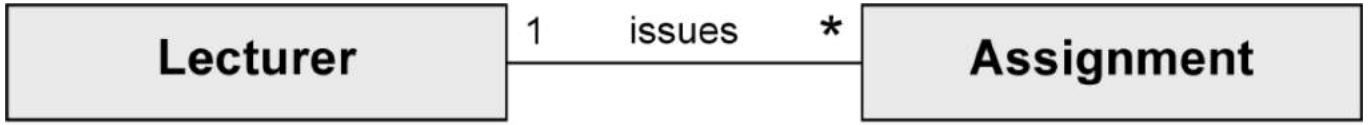
- Notasi Java-like :

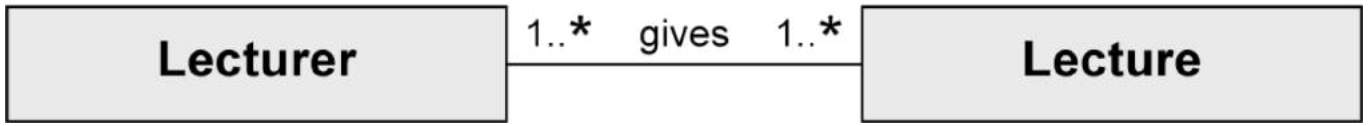
```
class Professor {…}

class Student{
   public Professor[] lecturer;
   …
}
```

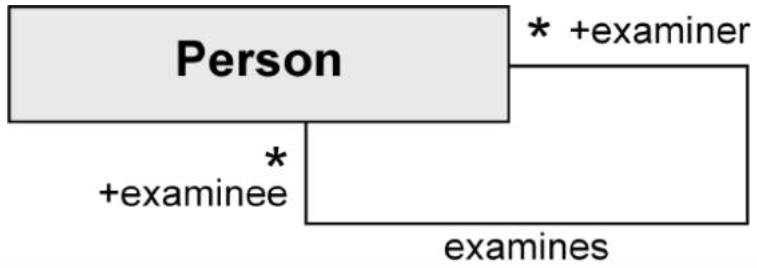
Binary Association – Multiplicity dan Role

- Multiplicity: jumlah object yang dapat diasosiasikan dengan satu object dari sisi lainnya



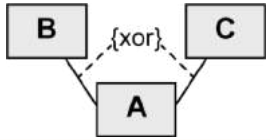


- Role: menggambarkan bagaimana sebuah object terlibat dalam association relationship

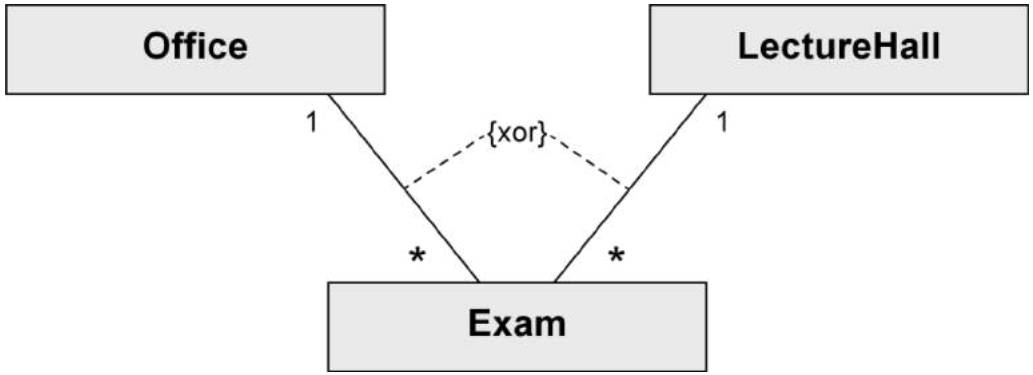


© Polibatam

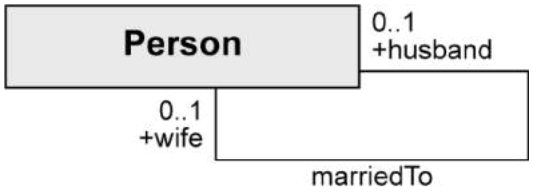
Binary Association – batasan `xor`

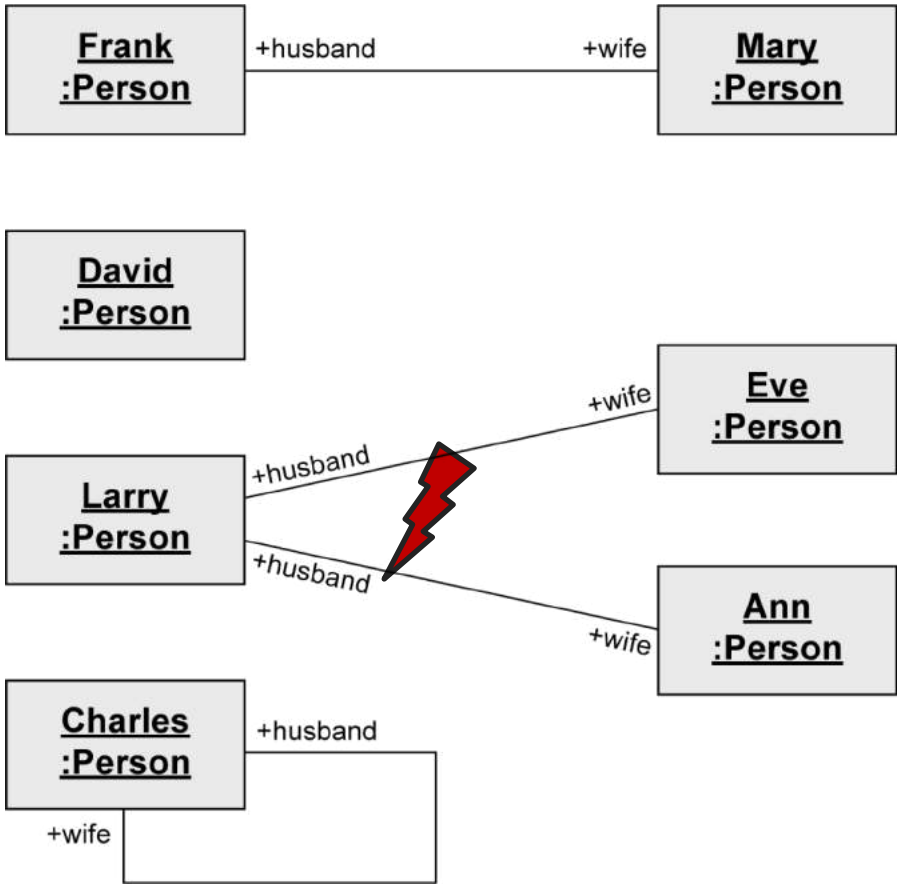


- Batasan “exclusive or”
- Sebuah object dari class **`A`** memiliki association dengan sebuah object dari class **`B`** atau sebuah object dari class **`C`** tapi tidak dengan keduanya.



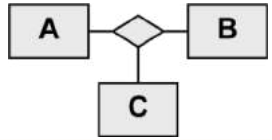
Unary Association - Contoh



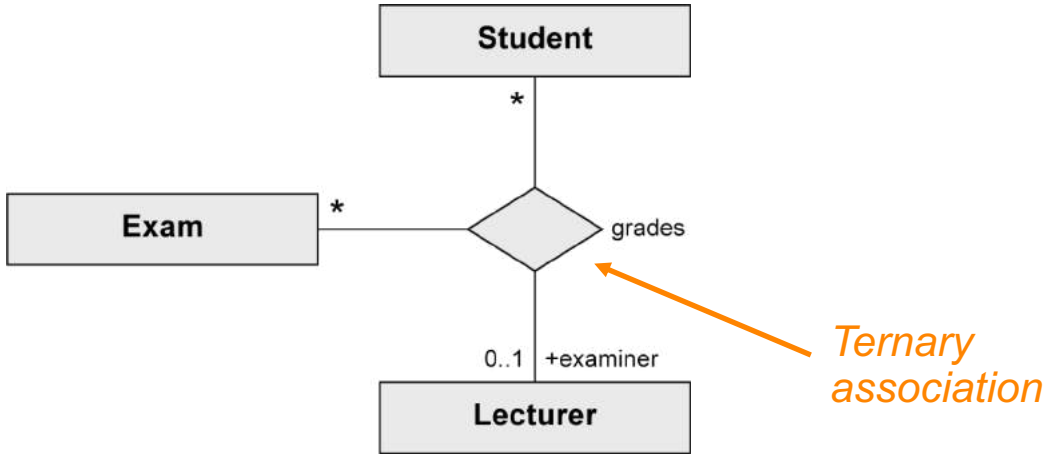


© Polibatam

n-ary Association (1/2)

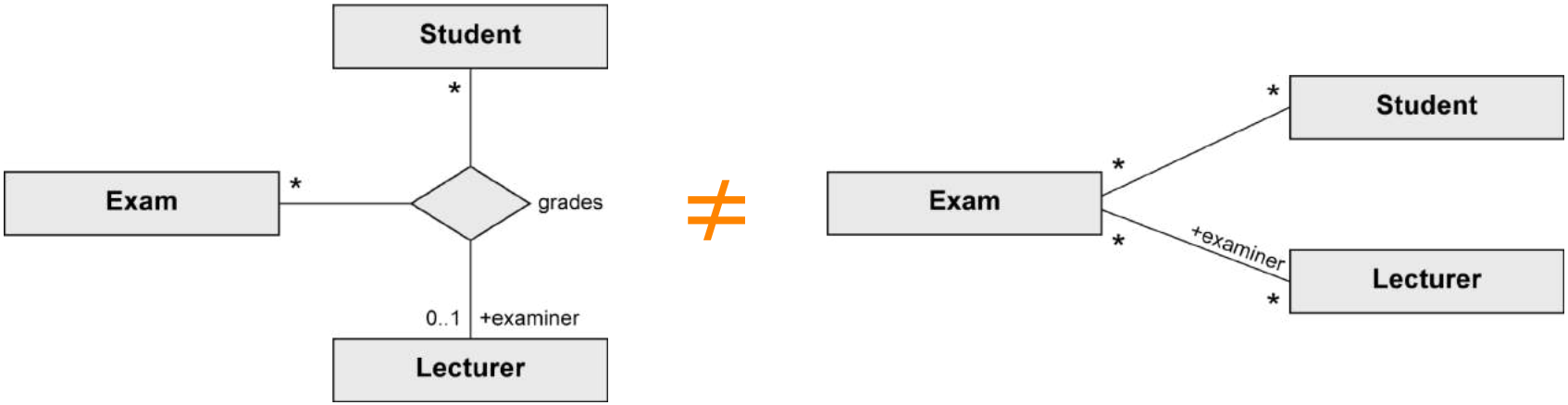


- Lebih dari dua object terlibat dalam sebuah relationship.
- Tidak ada arah navigation



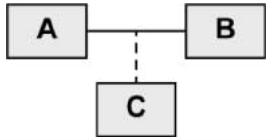
n-ary Association (2/2)

- Contoh
  - (**Student**, **Exam**) → (**Lecturer**)
    - Seorang ***student*** mengikuti sebuah ***exam*** dengan seorang atau tanpa ***lecturer***
  - (**Exam**, **Lecturer**) → (**Student**)
    - Sebuah ***exam*** dengan seorang ***lecturer*** dapat diikuti oleh beberapa ***students***
  - (**Student**, **Lecturer**) → (**Exam**)
    - Seorang ***student*** dapat dinilai oleh seorang ***Lecturer*** untuk beberapa ***exam***

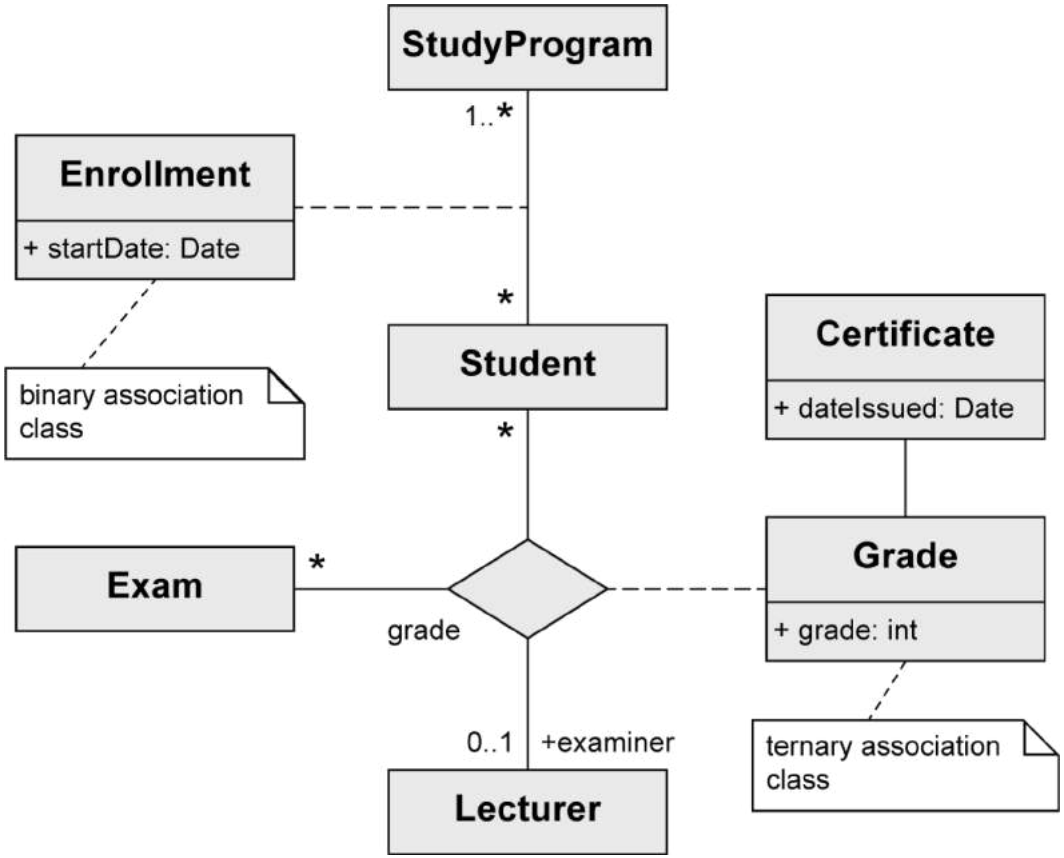


© Polibatam

Association Class

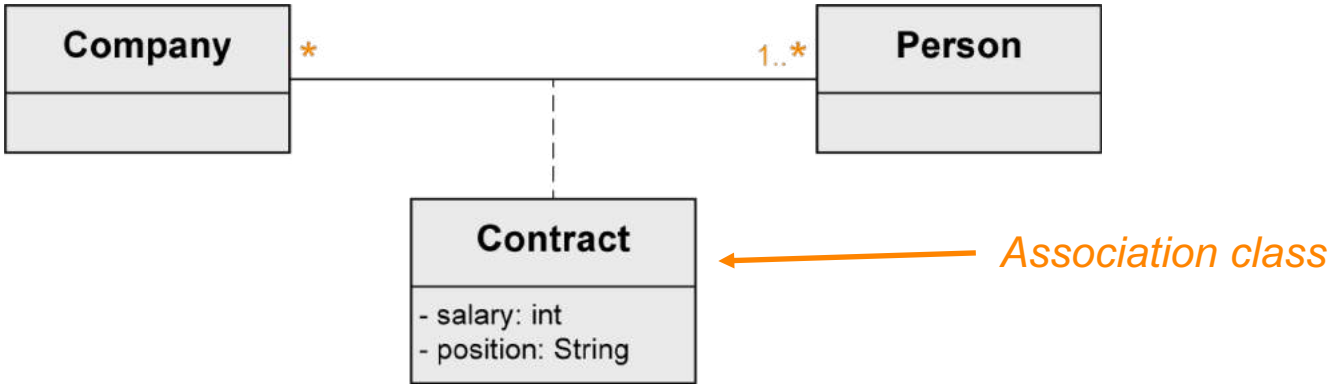


- Letakkan attribute ke relationship antar class daripada ke class itu sendiri

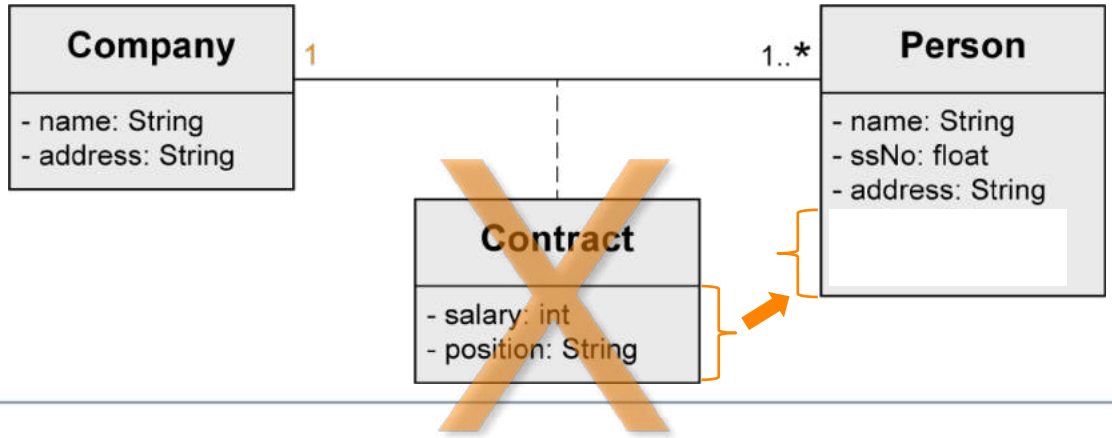


Association Class

- Diperlukan pada pemodelan Association n:m



- Dengan 1:1 atau 1:n dimungkinkan tapi tidak diperlukan



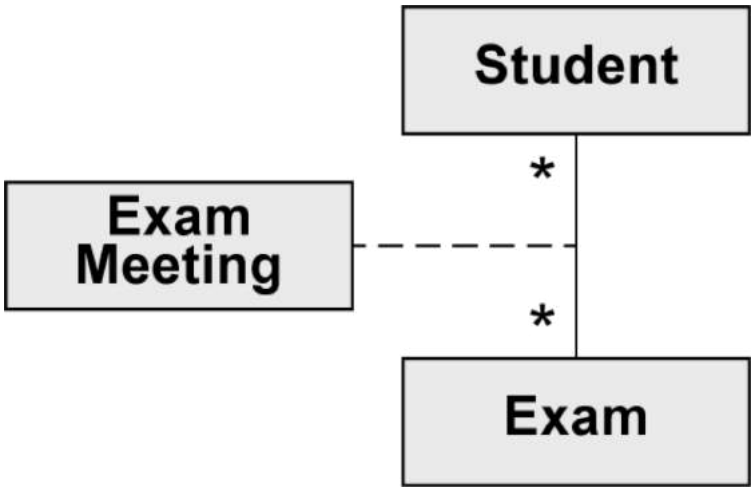
Association Class vs. Class Biasa

Seorang `Student` *dapat melakukan enroll* `untuk StudyProgram` *tertentu hanya sekali*

Seorang `Student` *dapat memiliki beberapa* `Enrollment` *untuk sebuah* `StudyProgram`

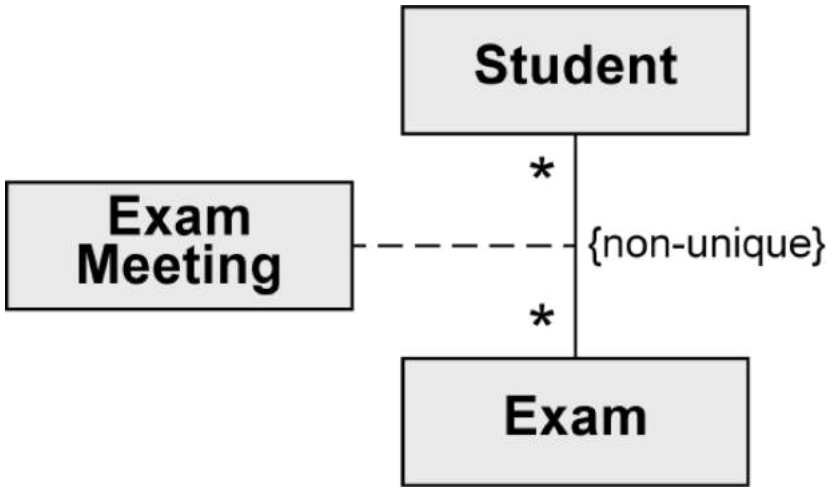
Association Class – `unique/non-unique` (1/2)

- Default: tidak ada duplikasi



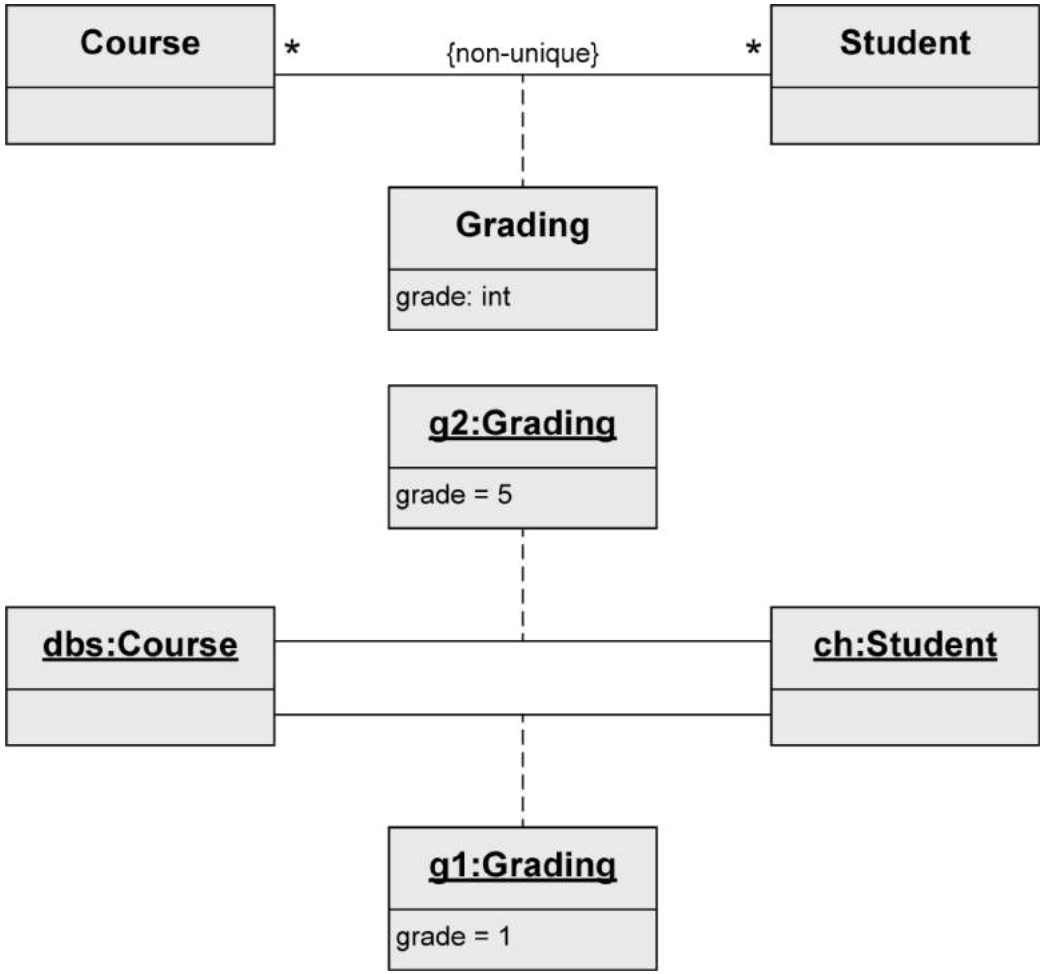
Seorang student hanya dapat melakukan exam meeting untuk sebuah exam sekali.

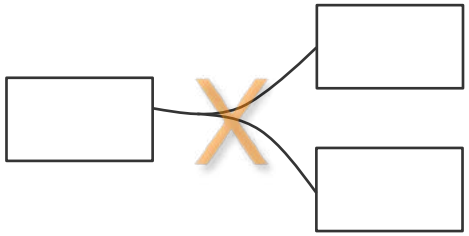
- non-unique: boleh duplikasi



Seorang student dapat melakukan lebih dari satu exam meetings untuk sebuah exam.

Association Class – `unique/non-unique` (2/2)



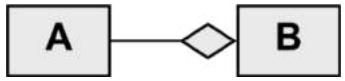


Aggregation

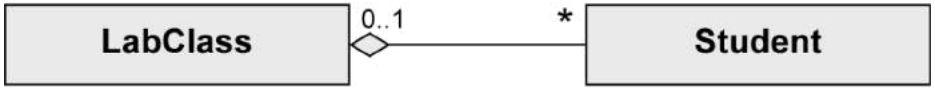
- Bentuk khusus association
- Digunakan untuk menggambarkan sebuah class merupakan bagian dari class lain
- Properti dari aggregation association:
  - **Transitive**: jika **`B`** adalah bagian dari **`A`** dan **`C`** adalah bagian dari **`B`**, **`C`** adalah bagian **`A`** juga
  - **Asymmetric**: tidak mungkin bahwa **`A`** adalah bagian dari **`B`** dan **`B`** adalah bagian dari **`A`** secara bersamaan.

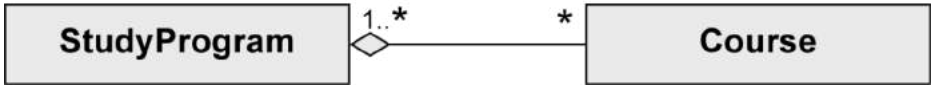
- Dua tipe:
  - Shared aggregation
  - Composition

Shared Aggregation



- Menggambarkan sebuah kepemilikan yang lemah
  = Bagian juga dapat berdiri sendiri
- Multiplicity pada ujung aggregating mungkin >1
  = Sebuah elemen dapat menjadi bagian dari beberapa elemen yang lain secara bersamaan
- Meliputi sebuah acyclic graph berarah
- Syntax: bentuk diamond berlubang pada ujung aggregating
- Contoh:
  - **`Student`** adalah bagian dari **`LabClass`**
  - **`Course`** adalah bagian dari **`StudyProgram`**





© Polibatam

Composition

- Keberadaan *dependency*/kebergantungan antara composite object dengan bagian-bagiannya
- Sebuah bagian hanya dapat menjadi bagian dari paling banyak satu composite object pada suatu saat

  Multiplicity di ujung aggregating max. 1

  -> Composite object membentuk sebuah *tree*/pohon
- Jika composite object dihapus, bagian-bagiannya juga ikut terhapus.
- Syntax: Bentuk diamond penuh pada ujung aggregating end
- Contoh: **`Beamer`** adalah bagian dari **`LectureHall`** adalah bagian dari **`Building`**

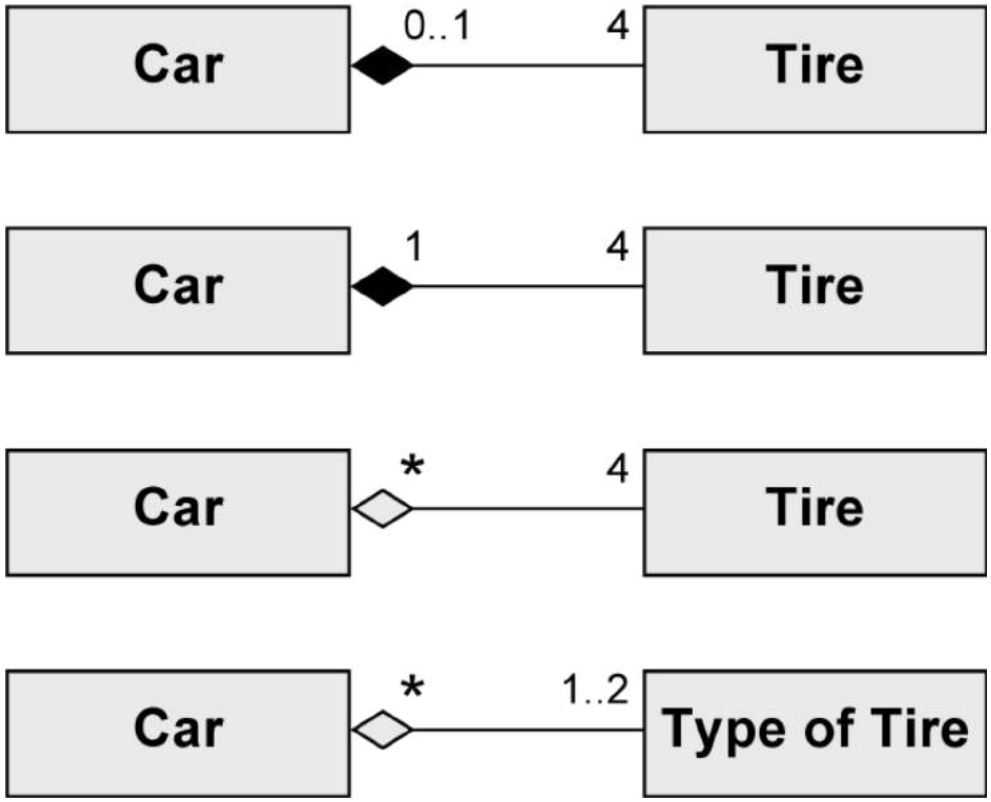
Jika `Building` dihapus, `LectureHall` ikut terhapus

`Beamer` dapat berdiri sendiri tanpa `LectureHall`, japi jika termasuk dalam `LectureHall` pada saat dihapus, maka `Beamer` ikut terhapus

© Polibatam

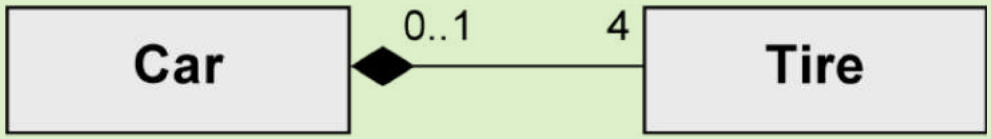
Shared Aggregation dan Composition

- Model mana yang dapat diterapkan?



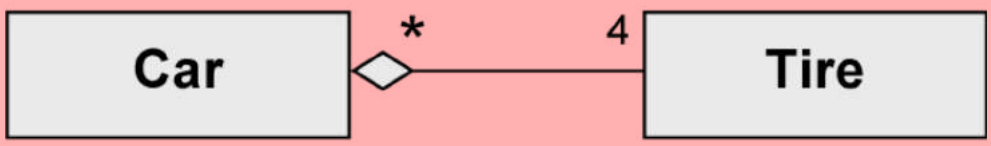
Shared Aggregation dan Composition

- Model mana yang dapat diterapkan?

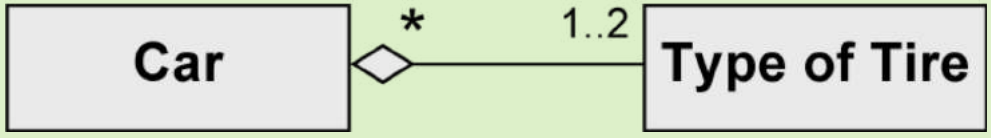


Sebuah `Tire` dapat berdiri sendiri tanpa `Car`. Sebuah `Tire` terdapat pada sebuah `Car` saja. — Yes

Sebuah `Tire` tidak ada tanpa ada `Car`. — No

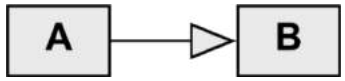


Sebuah `Tire` dapat digunakan di beberapa `Car` pada waktu yg sama. — No

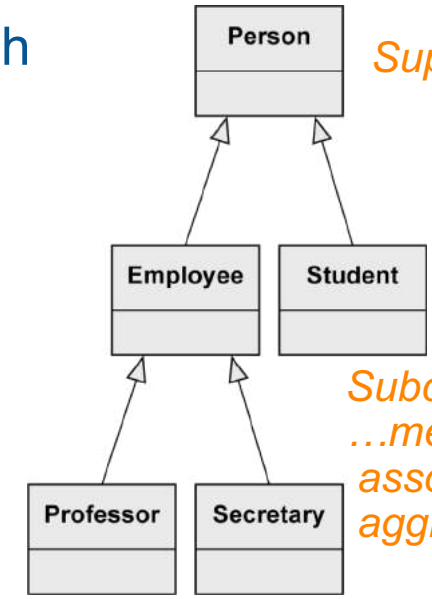


Sebuah `Car` memiliki satu atau dua type `Tire`. Beberapa `Car` mungkin memiliki Type of `Tires` yang sama. — Yes

Generalization



- Karakteristik (attribute dan operation), association, and aggregation milik sebuah general class (superclass) diteruskan ke semua subclass-nya.
- Setiap instance of dari sebuah subclass secara tidak langsung juga merupakan instance dari superclass.
- Subclass mewarisi semua karakteristik, association, dan aggregation dari superclass kecuali yang private.
- Subclass mungkin punya karakteristik, association, dan aggregation lain.
- Generalizations bersifat transitive.

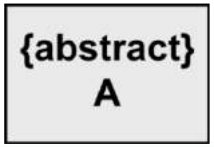


Superclass

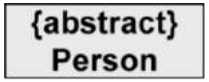
Subclasses …mewarisi karakteristik, association, dan aggregation

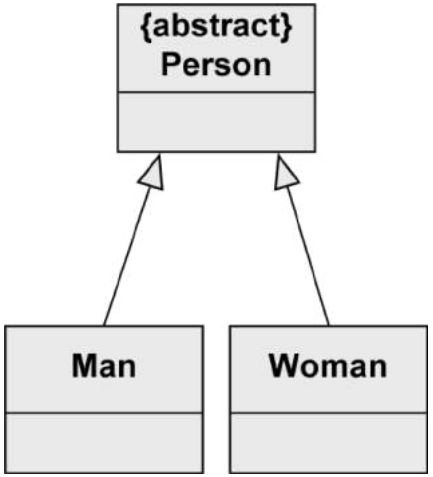
Seorang `Secretary` adalah seorang `Employee` dan seorang `Person`

Generalization – Abstract Class



- Digunakan untuk menunjukkan karakteristik yang sama dari subclass.
- Digunakan untuk menjamin bahwa tidak ada instance langung dari superclass.
- Hanya subclasse yang non-abstract yang bisa dinstansiasi.
- Berguna dalam konteks generalization relationships.
- Notasi: keyword **`{abstract}`** atau nama class dimiringkan





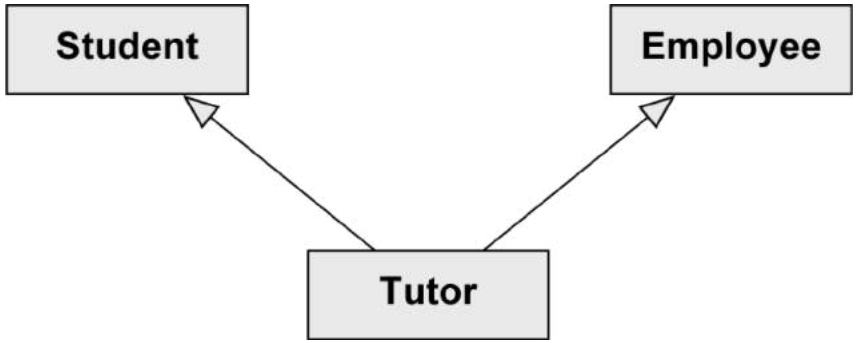
Tidak mungkin ada object `Person`

Dua tipe dari `Person`: `Man` dan `Woman`

Generalization – Multiple Inheritance

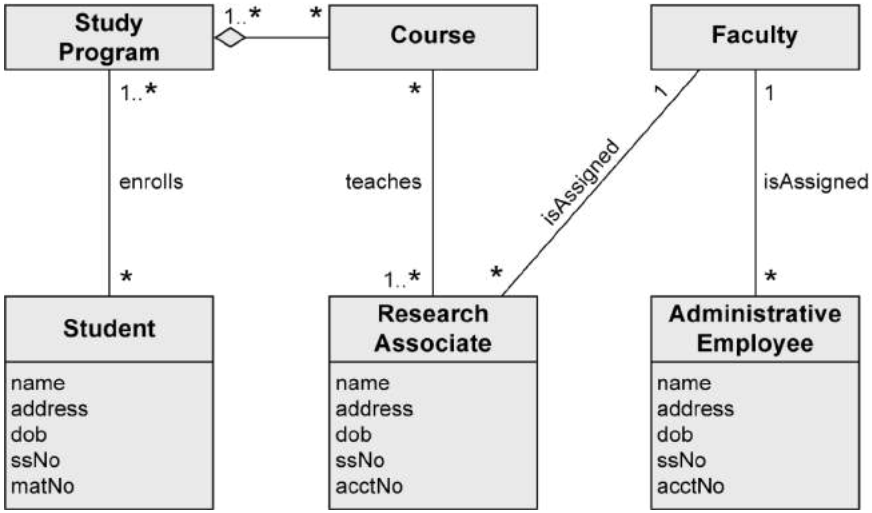
- UML memungkinkan adanya multiple inheritance.
- Sebuah class mungkin memiliki beberapa superclass.

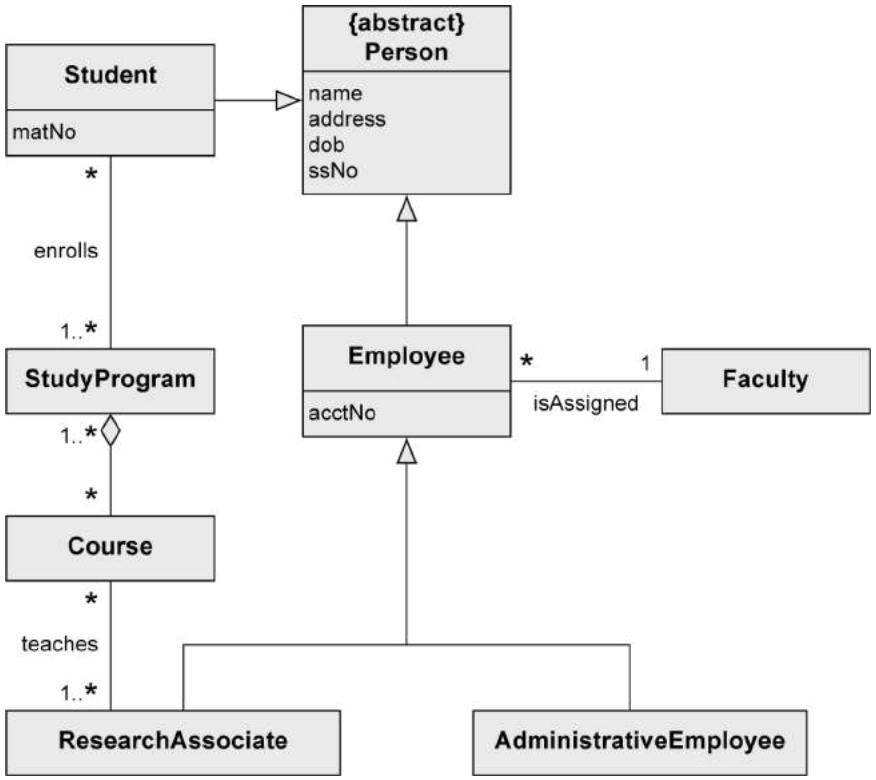
- Contoh:



Seorang `Tutor` adalah seorang `Employee` dan seorang `Student`

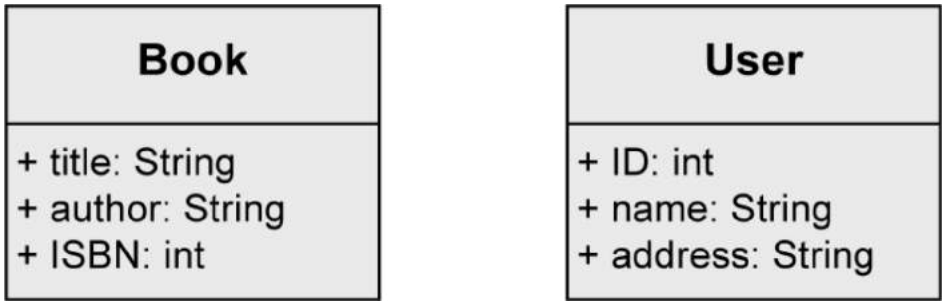
Dengan dan Tanpa Generalization





Membangun Class Diagram

- Tidak mungkin untuk mendapatkan class, attribute dan association dari sebuah teks dengan natural language secara otomatis.
- Guidelines
  - Kata benda seringkali mengindikasikan class
  - Kata sifat mengindikasikan nilai attribute
  - Kata kerja mengindikasikan operation
- Contoh: *Library management system* menyimpan data user dengan ID yang unik, name dan address, serta data book dengan title, author dan ISBN. Ann Foster ingin menggunakan perpustakaan/*library*.



© Polibatam

Pertanyaan: Bagaimana dengan Ann Foster?

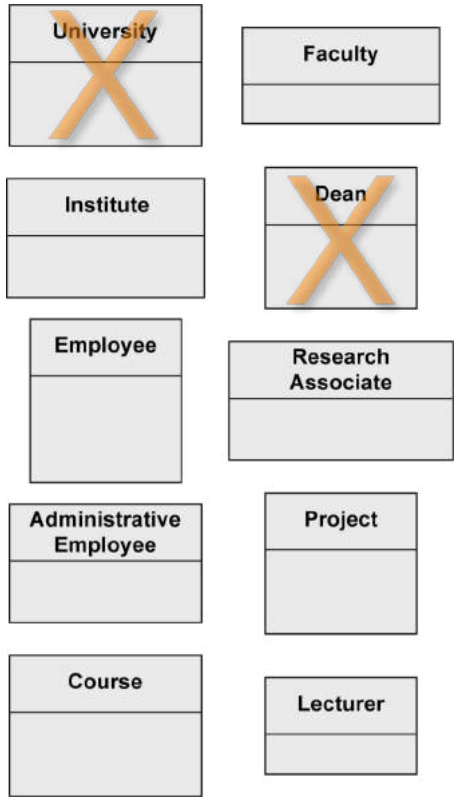
Contoh – University Information System

- Sebuah university terdiri dari beberapa faculty yang melingkupi beberapa institute. Setiap faculty dan setiap institute memiliki name/nama. Address/alamat dimiliki oleh setiap institute.
- Setiap faculty dipimpin oleh seorang dean, yang merupakan seorang employee dari university.
- Jumlah total employee tidak diketahui. Employee memiliki social security number, sebuah name/nama, dan sebuah email address. Terdapat perbedaan antara personnel research dan administrative.
- Research associate ditugaskan pada minimal satu institute. Field of study atau bidang ilmu setiap research associate diketahui. Research associate dapat terlibat pada project untuk sejumlah jam/hours, dan name/nama, starting date, dan end date dari projects diketahui. Beberapa research associate mengadakan course. Sehingga mereka disebut lecturer.
- Course memiliki ID (nomor unik), sebuah name/nama, dan weekly duration/durasi mingguan dalam jam.

Contoh – Langkah 1: Mengidentifikasi Class

- Sebuah university terdiri dari beberapa faculty yang melingkupi beberapa institute. Setiap faculty dan setiap institute memiliki name/nama. Address/alamat dimiliki oleh setiap institute.
- Setiap faculty dipimpin oleh seorang dean, yang merupakan seorang employee dari university.
- Jumlah total employee tidak diketahui. Employee memiliki social security number, sebuah name/nama, dan sebuah email address. Terdapat perbedaan antara personnel research dan administrative.
- Research associate ditugaskan pada minimal satu institute. Field of study atau bidang ilmu setiap research associate diketahui. Research associate dapat terlibat pada project untuk sejumlah jam/hours, dan name/nama, starting date, dan end date dari projects diketahui. Beberapa research associate mengadakan course. Sehingga mereka disebut lecturer.
- Course memiliki ID (nomor unik), sebuah name/nama, dan weekly duration/durasi mingguan dalam jam.

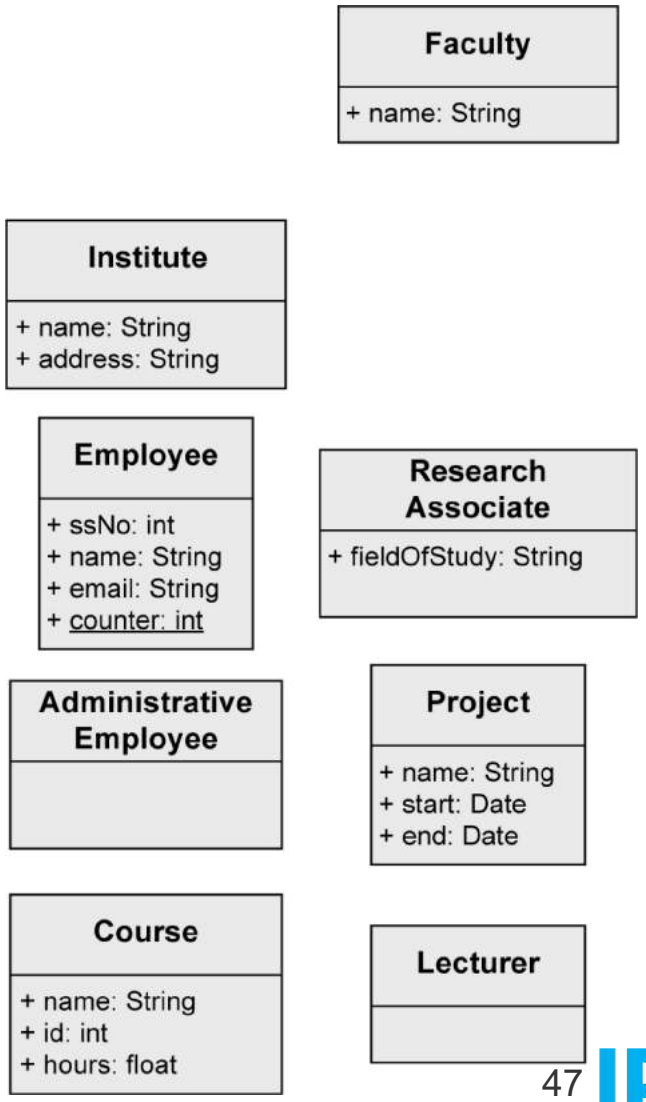
Kita memodelkan sistem „University“



Dean tidak memiliki attribute selain attribute milik employee

Contoh – Langkah 2: Mengidentifikasi Attribute

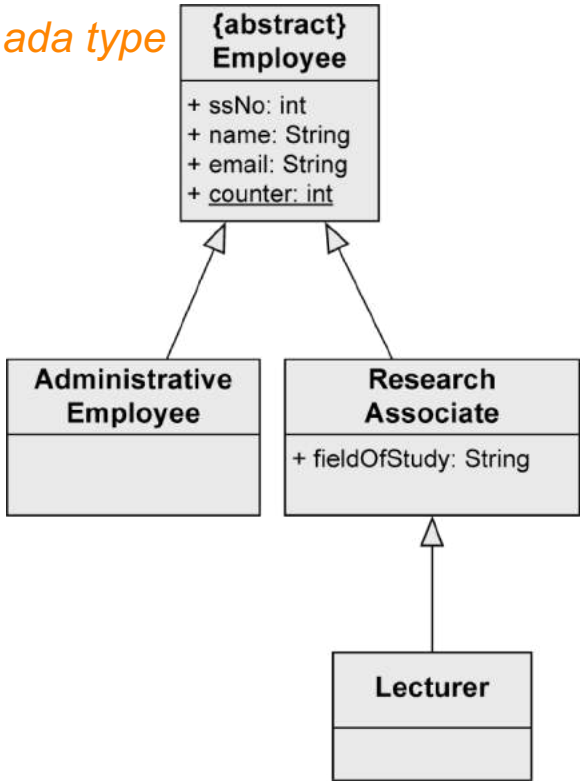
- Sebuah university terdiri dari beberapa faculty yang melingkupi beberapa institute. Setiap faculty dan setiap institute memiliki name/nama. Address/alamat dimiliki oleh setiap institute.
- Setiap faculty dipimpin oleh seorang dean, yang merupakan seorang employee dari university.
- Jumlah total employee tidak diketahui. Employee memiliki social security number, sebuah name/nama, dan sebuah email address. Terdapat perbedaan antara personnel research dan administrative.
- Research associate ditugaskan pada minimal satu institute. Field of study atau bidang ilmu setiap research associate diketahui. Research associate dapat terlibat pada project untuk sejumlah jam/hours, dan name/nama, starting date, dan end date dari projects diketahui. Beberapa research associate mengadakan course. Sehingga mereka disebut lecturer.
- Course memiliki ID (nomor unik), sebuah name/nama, dan weekly duration/durasi mingguan dalam jam.



Contoh – Langkah 2: Mengidentifikasi Relationship (1/6)

- Tiga jenis relationship:
  - Association
  - Generalization
  - Aggregation

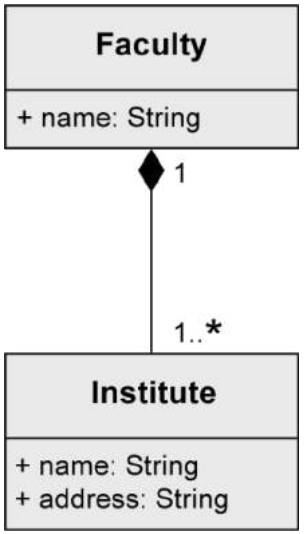
Abstract, yaitu tidak ada type lain dari employees



- Indikasi sebuah generalization
- *"Terdapat perbedaan antara personnel research dan administrative."*
- *"Beberapa research associate mengadakan course. Sehingga mereka disebut lecturer."*

Contoh – Langkah 2: Mengidentifikasi Relationship (2/6)

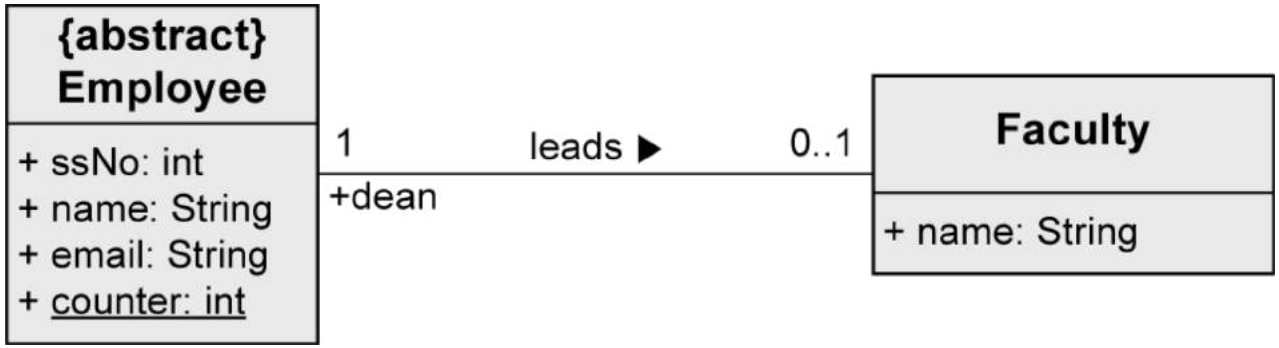
- *"Sebuah university terdiri dari beberapa faculty yang melingkupi beberapa institute."*



Composition untuk menunjukkan adanya dependency/kebergantungan

Contoh – Langkah 2: Mengidentifikasi Relationship (3/6)

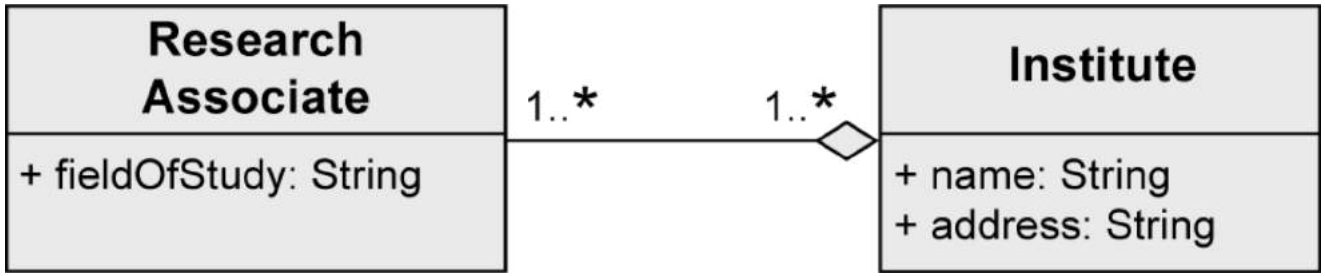
- *"Setiap faculty dipimpin oleh seorang dean, yang merupakan seorang employee dari university."*



Dalam `leads`-relationship,
`Employee` memiliki role sebagai seorang `dean`.

Contoh – Langkah 2: Mengidentifikasi Relationship (4/6)

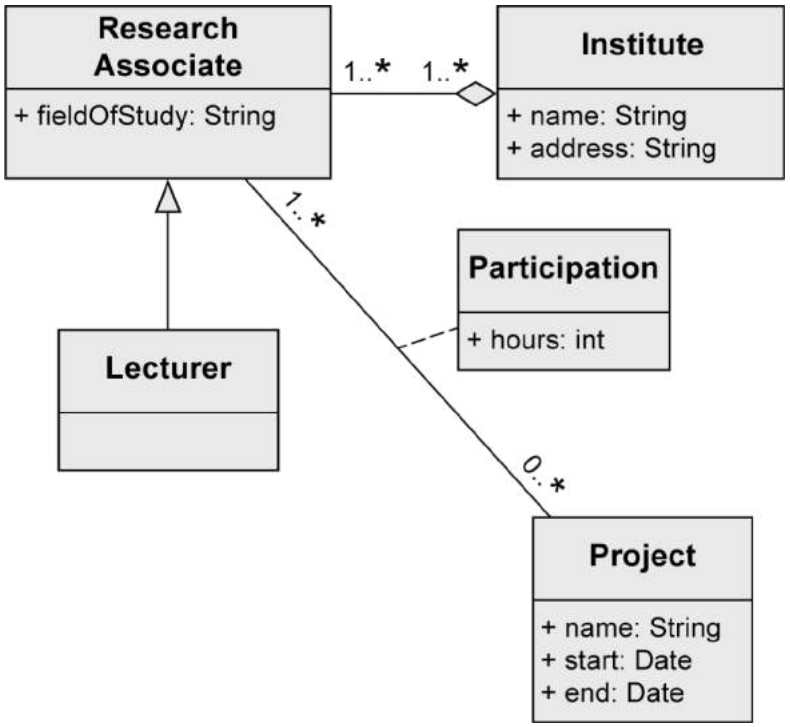
- "Research associate ditugaskan pada minimal satu institute."



Shared aggregation untuk menunjukkan bahwa `ResearchAssociates` merupakan bagian dari sebuah `Institute`, tapi tidak ada existence dependency

Contoh – Langkah 2: Mengidentifikasi Relationship (5/6)

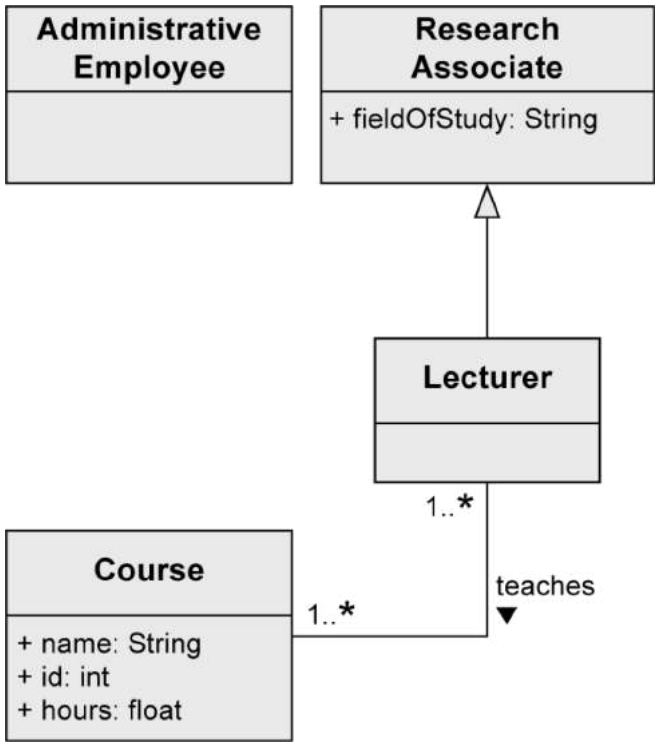
- *"Research associate dapat terlibat pada project untuk sejumlah jam/hours"*



Association class memungkinkan untuk mencatat jumlah hours untuk setiap `Project` *oleh setiap* `ResearchAssociate`

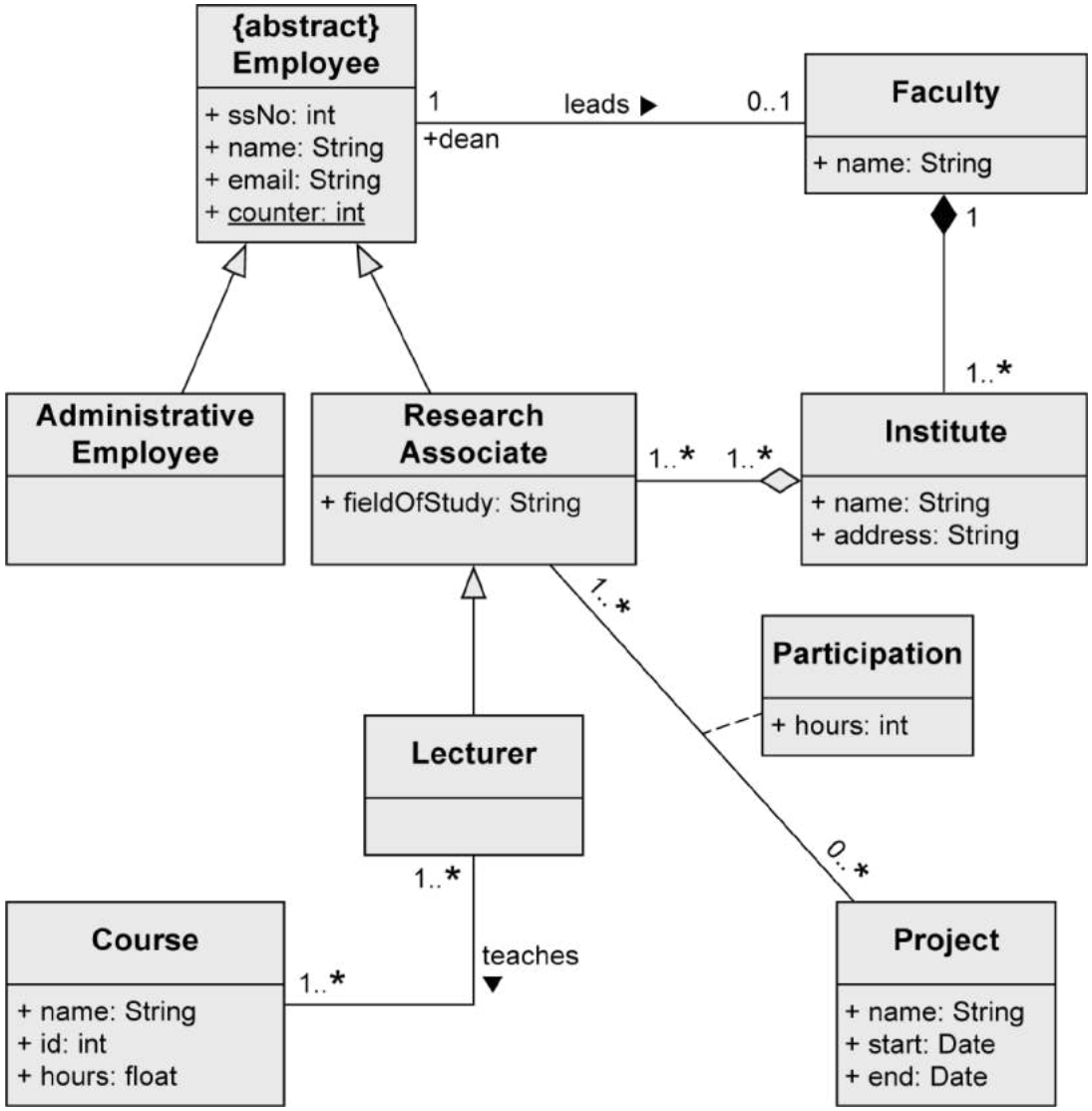
Contoh – Langkah 2: Mengidentifikasi Relationship (6/6)

- *"Beberapa research associate mengadakan course. Sehingga mereka disebut lecturer."*



`Lecturer` mewarisi semua karakteristik, association, dan aggregation dari `ResearchAssociate`.
Sebagai tambahan, seorang `Lecturer` memiliki association `teaches` dengan `Course`.

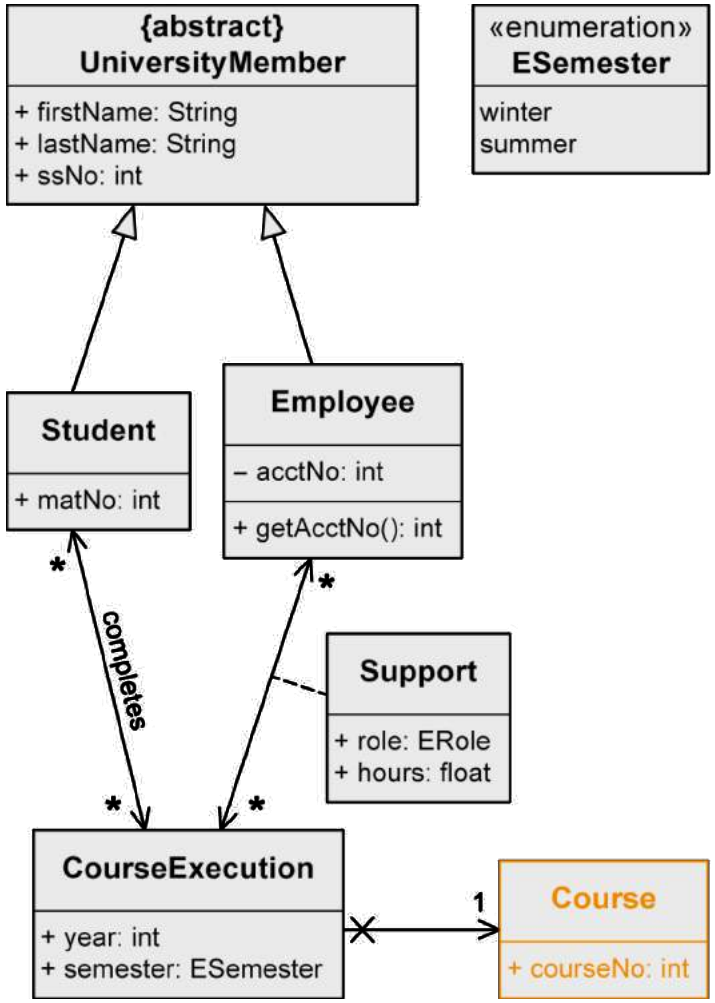
Contoh – Class Diagram Lengkap



Code Generation

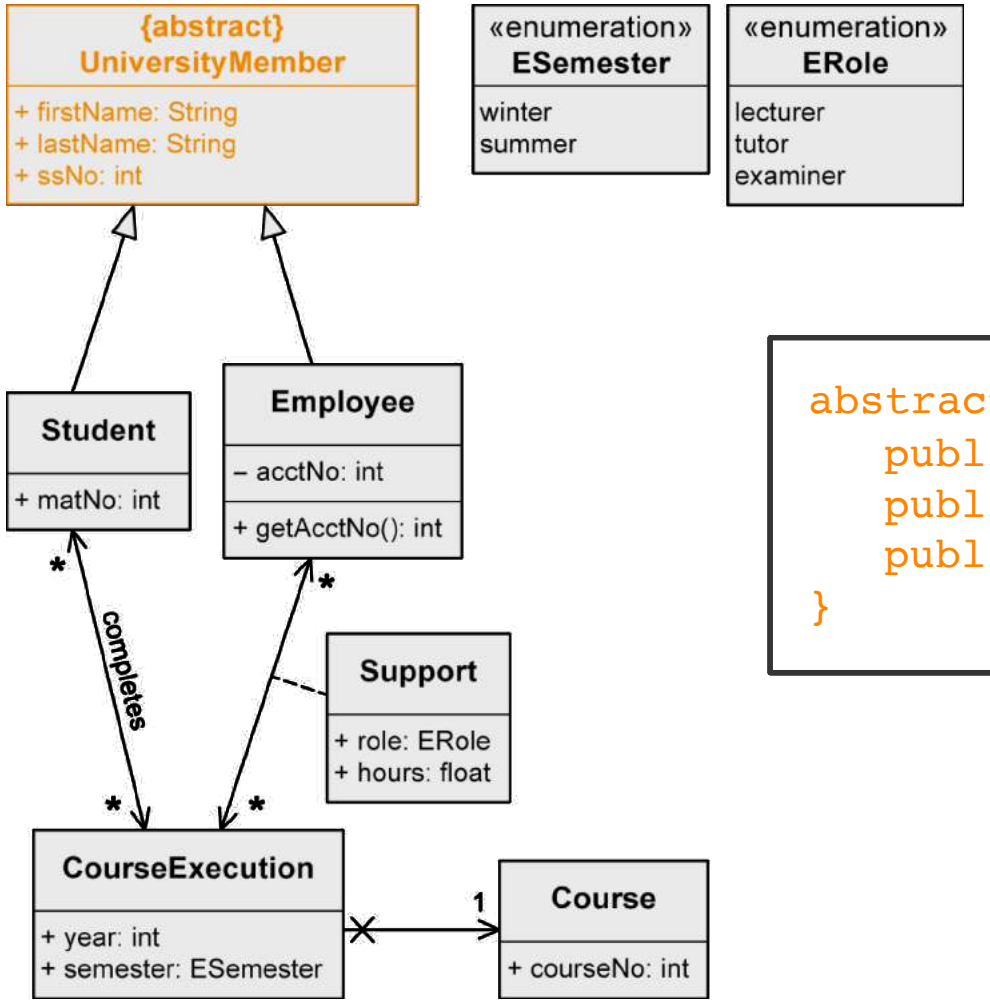
- Class diagram seringkali dibangun dengan maksud untuk mengimplementasikan elemen-elemen yang dimodelkan tersebut dalam sebuah bahasa pemrograman berorientasi object (*object-oriented programming language*).
- Seringkali, penerjemahan dilakukan secara semi-automatic dan hanya memerlukan sedikit bantuan manual.

Code Generation – Contoh (1/6)




```
class Course {
    public int courseNo;
}
```

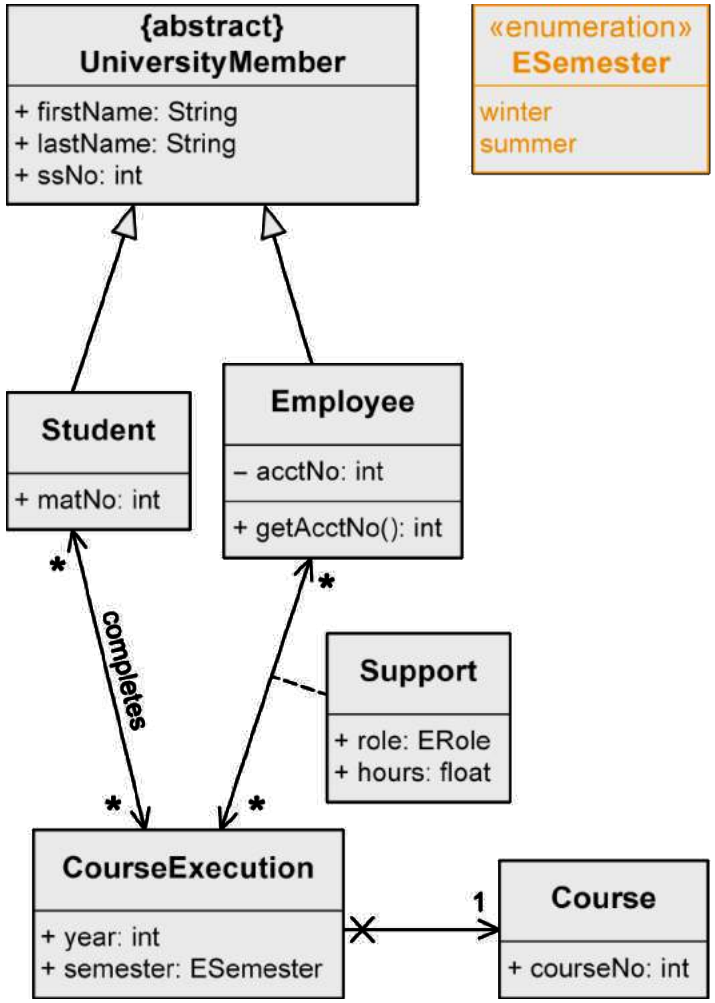
Code Generation – Contoh (2/6)




```
abstract class UniversityMember {
   public String firstName;
   public String lastName;
   public int ssNo;
}
```

© Polibatam

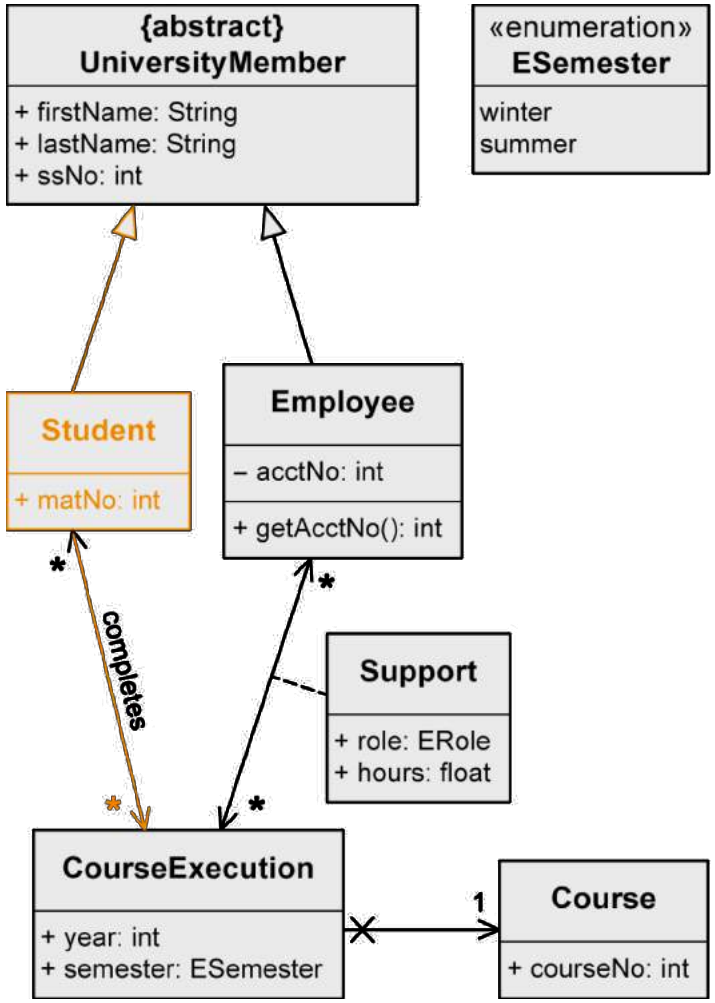
Code Generation – Contoh (3/6)




```
Enumeration ESemester {
    winter,
    summer
}
```

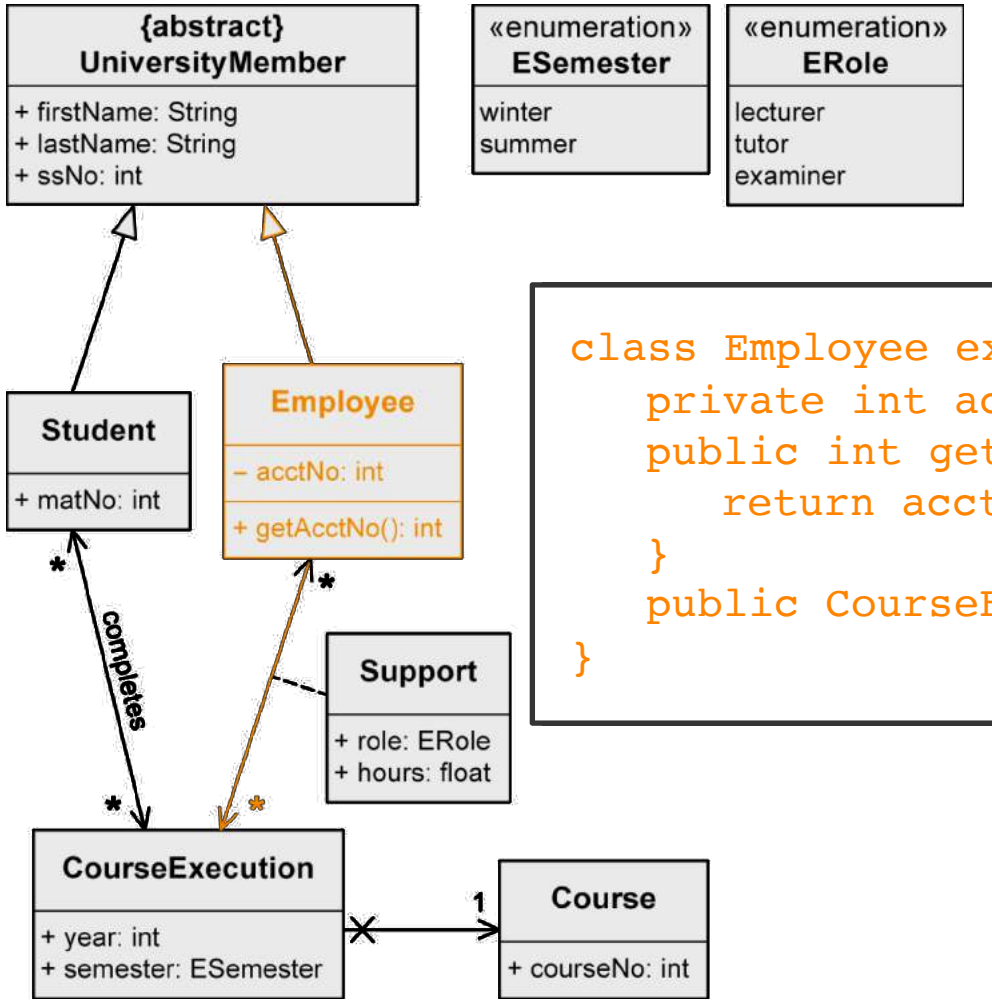
```
Enumeration ERole {
    lecturer,
    tutor,
    examiner
}
```

Code Generation – Contoh (4/6)




```
class Student extends
                UniversityMember {
   public int matNo;
   public CourseExecution []
                completedCourses;
}
```

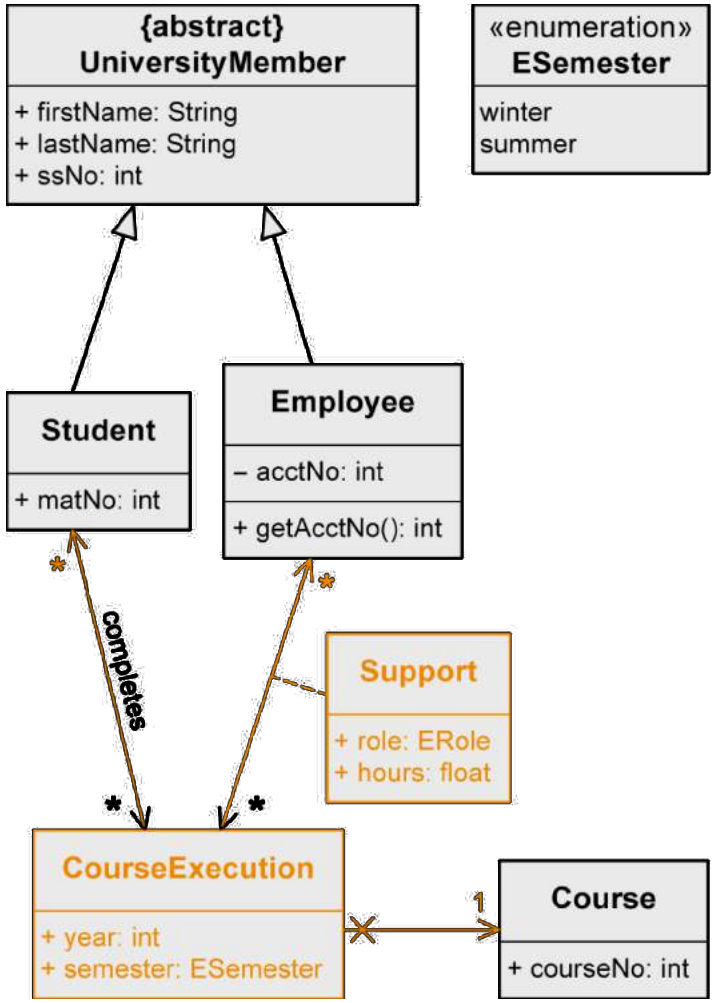
Code Generation – Contoh (5/6)




```
class Employee extends UniversityMember {
   private int acctNo;
   public int getAcctNo () {
      return acctNo;
   }
   public CourseExecution [] courseExecutions;
}
```

© Polibatam

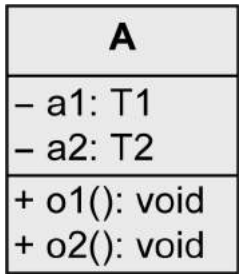
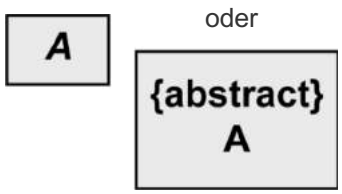
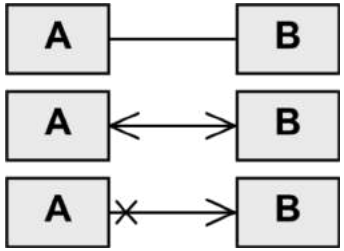
Code Generation – Contoh (6/6)



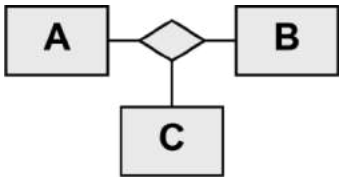
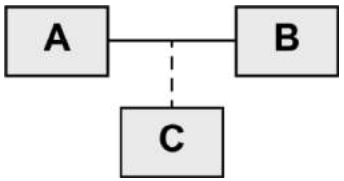
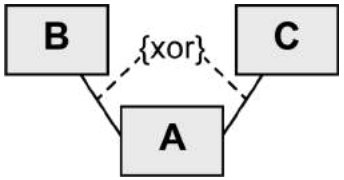

```
class CourseExecution {
   public int year;
   public ESemester semester;
   public Student [] student;
   public Course course;
   public Hashtable support;
      // Key: employee
      // Value: (role, hours)
}
```

© Polibatam

Notasi Elemen (1/3)

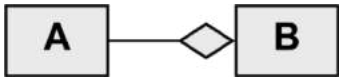
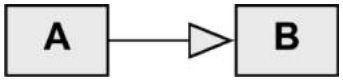
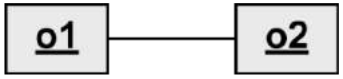
Name	Notation	Description
Class		Deskripsi struktur dan perilaku dari kumpulan object
Abstract class		Class yang tidak dapat diinstansiasi
Association		Relationship antar class: navigability tidak ditentukan, navigability dua arah, tidak ada navigability di salah satu arah

Notasi Elemen (2/3)

Name	Notation	Description
n-ary association		Relationship antara n (disini 3) class
Association class		Deskripsi lebih detail dari sebuah association
xor relationship		Sebuah object dari **C** memiliki relationship dengan sebuah object dari **A** atau sebuah object dari **B** tapi tidak keduanya

© Polibatam

Notasi Elemen (3/3)

Name	Notation	Description
Shared aggregation		Relationship yang melambangkan bagian-dari (**A** bagian dari **B**)
Strong aggregation = composition		Existence-dependent relationship yang melambangkan bagian-dari (**A** bagian dari **B**)
Generalization		Relationship pewarisan (**A** mewarisi dari **B**)
Object		Instance dari sebuah class
Link		Relationship antar objects