UML Class Diagrams part 1

Idaho State University | Computer Science

Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

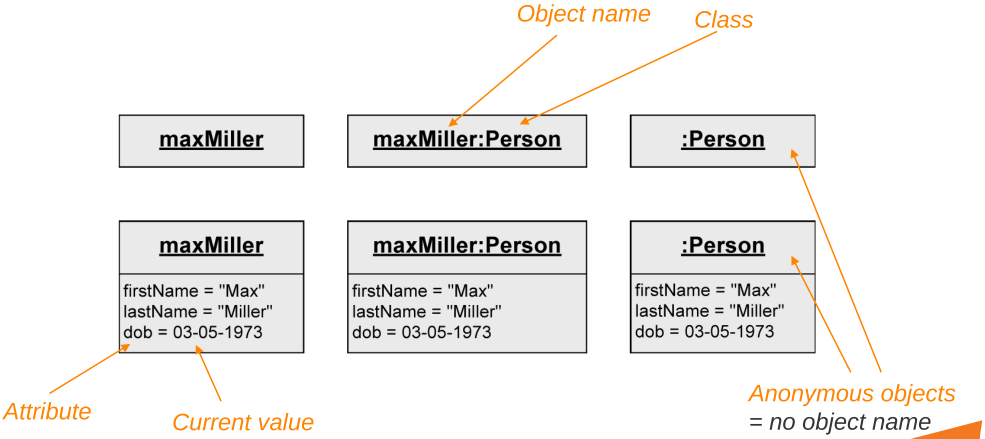
ROAR

Outline

- UML Class Diagrams
- UML Object Diagrams

Object

- Individuals of a system
- Alternative notations:



Object name

Class

maxMiller

maxMiller:Person

:Person

maxMiller
firstName = "Max"
lastName = "Miller"
dob = 03-05-1973

maxMiller:Person
firstName = "Max"
lastName = "Miller"
dob = 03-05-1973

:Person
firstName = "Max"
lastName = "Miller"
dob = 03-05-1973

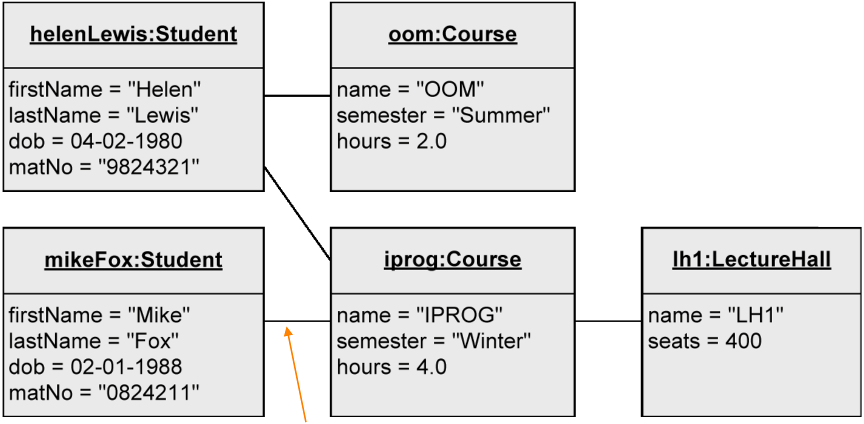
Attribute

Current value

Anonymous objects
= no object name

Object Diagrams

- Objects of a system and their relationships (links)
- Snapshot of objects at a specific moment in time



helenLewis:Student

firstName = "Helen"
lastName = "Lewis"
dob = 04-02-1980
matNo = "9824321"

oom:Course

name = "OOM"
semester = "Summer"
hours = 2.0

mikeFox:Student

firstName = "Mike"
lastName = "Fox"
dob = 02-01-1988
matNo = "0824211"

iprog:Course

name = "IPROG"
semester = "Winter"
hours = 4.0

lh1:LectureHall

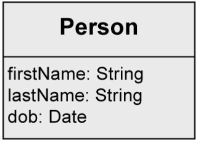
name = "LH1"
seats = 400

Link

From Object to Class

- Individuals of a system often have identical characteristics and behavior
- A class is a construction plan for a set of similar objects of a system
- Objects are instances of Classes
- Attributes; structural characteristics of a class
  - Different value for each instance (= object)
- Operations: behavior of a class
  - Identical for all objects of a class (not depicted in object diagram)

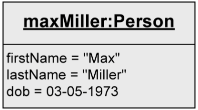
Class



Person

firstName: String
lastName: String
dob: Date

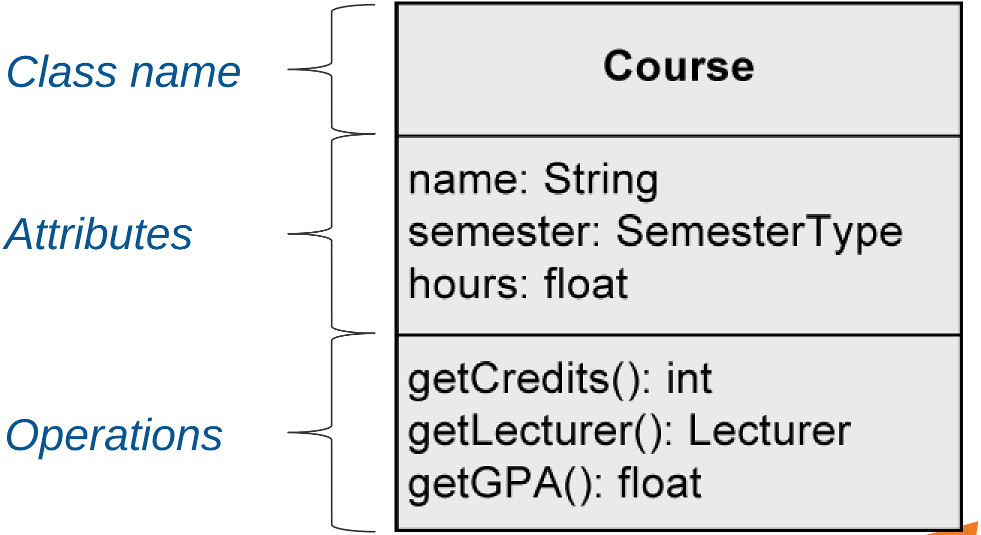
Object of that class



<u>maxMiller:Person</u>

firstName = "Max"
lastName = "Miller"
dob = 03-05-1973

Class



Class name

Attributes

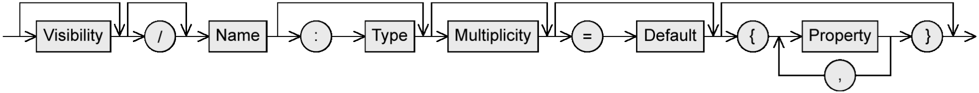
Operations

Course

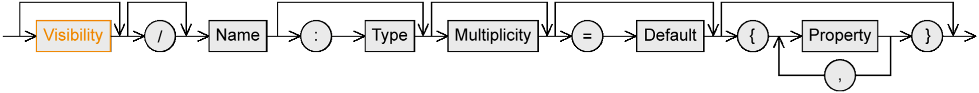
name: String
semester: SemesterType
hours: float

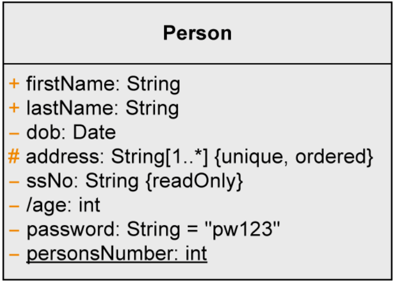
getCredits(): int
getLecturer(): Lecturer
getGPA(): float

Attribute Syntax



Attribute Syntax - Visibility



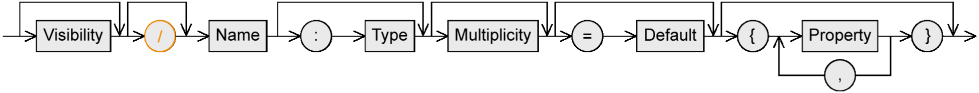


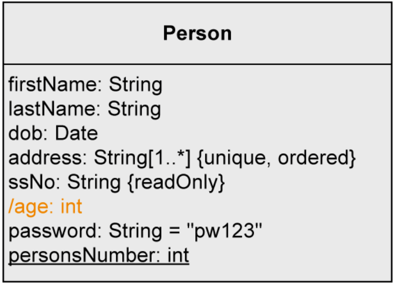
Person

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

- Who is permitted to access the attribute
  - + … public: everybody
  - - … private: only the object itself
  - # … protected: class itself and subclasses
  - ~ … package: classes that are in the same package

Attribute Syntax - Derived Attributes



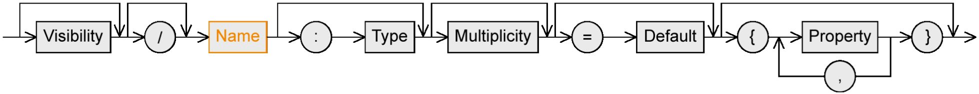


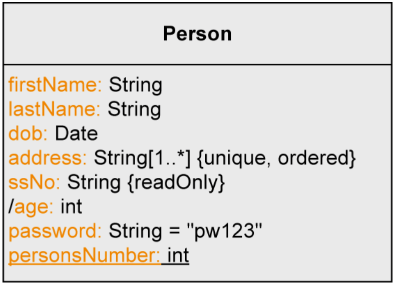
Person

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

- Attribute value is derived from other attrib utes
  - **age**: calculated from the date of birth

Attribute Syntax - Name



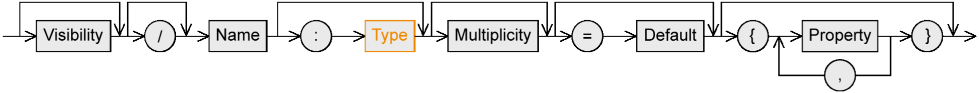

Person

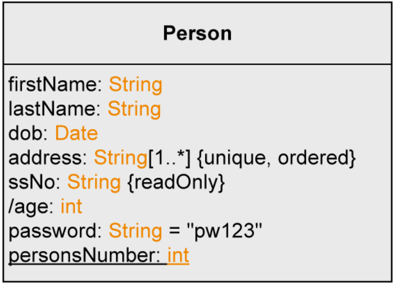
firstName: String
lastName: String
dob: Date
address: String[1..*] {unique, ordered}
ssNo: String {readOnly}
/age: int
password: String = "pw123"
personsNumber: int

- Name of the attribute

Idaho State University
Computer Science

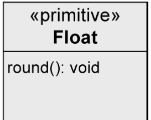
Attribute Syntax - Type



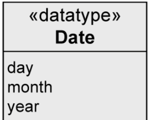

Person

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

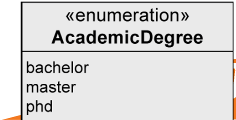
- Type
  - User-defined classes
  - Data type
    - Pre-defined: Boolean, Integer, UnlimitedNatural, String
    - User-defined: ≪primitive≫
    - Composite data type: ≪datatype≫
    - Enumerations: ≪enumeration≫


«primitive»
Float

round(): void

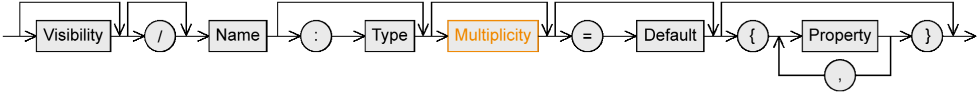

«datatype»
Date

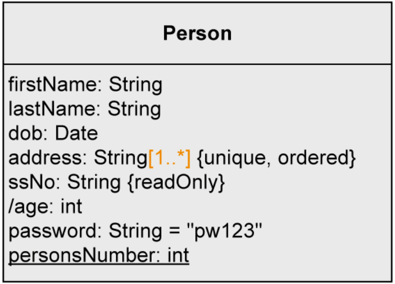
day
month
year


«enumeration»
AcademicDegree

bachelor
master
phd

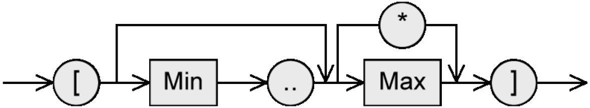
Attribute Syntax - Multiplicity



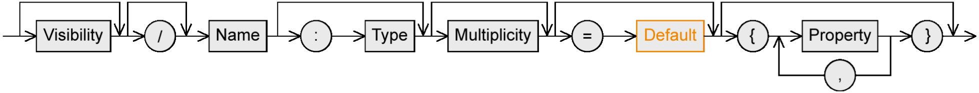


firstName: String
lastName: String
dob: Date
address: String[1..*] {unique, ordered}
ssNo: String {readOnly}
/age: int
password: String = "pw123"
personsNumber: int

- Number of values an attribute may contain
- Default value: 1
- Notation: **[min..max]**
  - no upper limit: [*] or [0..*]



Attribute Syntax - Default Value

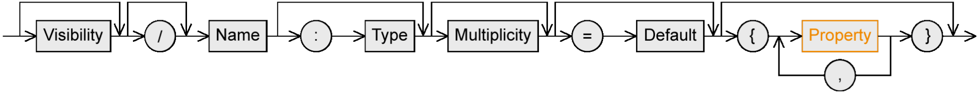


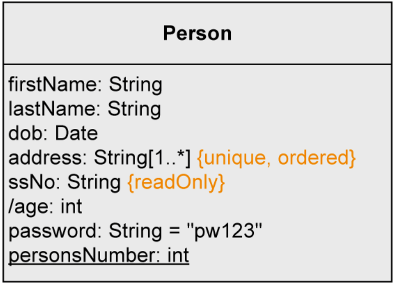
Person

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

- Default value
  - Used if the attribute value is not set explicitly by the user

Attribute Syntax - Properties



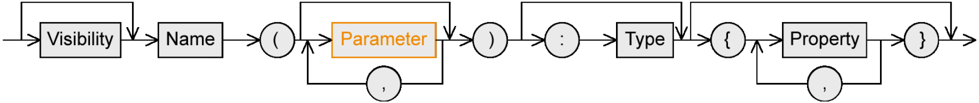


Person

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

- Pre-defined Properties
  - `{readOnly}` … value cannot be changed
  - `{unique}` … no duplicates permitted
  - `{non-unique}` … duplicates permitted
  - `{ordered}` … fixed order of the values
  - `{unordered}` … no fixed order to the values
- Attribute specification
  - Set: `{unordered, unique}`
  - Multi-set: `{unordered, non-unique}`
  - Ordered set: `{ordered, unique}`
  - List: `{ordered, non-unique}`

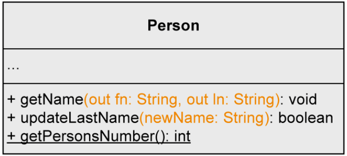
Operation Syntax - Parameters



Visibility → Name → (→ Parameter →) → : → Type → { → Property → }

,

,

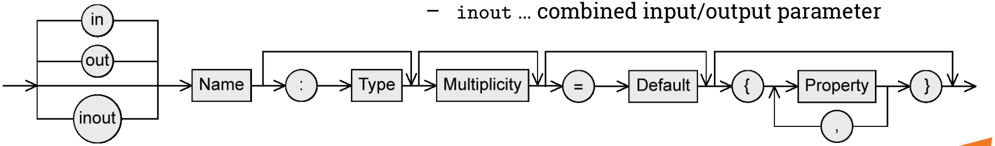


Person

...

+ getName(out fn: String, out ln: String): void
+ updateLastName(newName: String): boolean
<u>+ getPersonsNumber(): int</u>

- Notation similar to attributes
- Direction of the parameter
  - `in` … input parameter
    - When the operation is used, a value is expected from this parameter
  - `out` … output parameter
    - After the execution of the operation, the parameter has adopted a new value
  - `inout` … combined input/output parameter



in

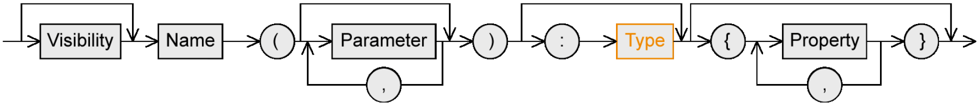
out

inout

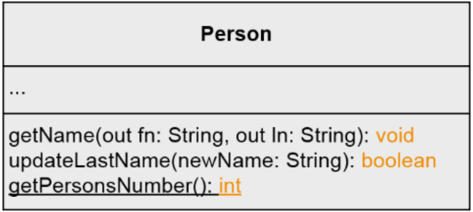
Name → : → Type → Multiplicity → = → Default → { → Property → }

,

Operation Syntax - Type



Visibility Name (Parameter ,) : Type { Property , }



Person

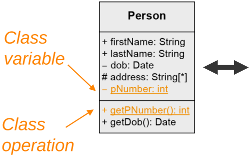
...

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

- Type of the return value

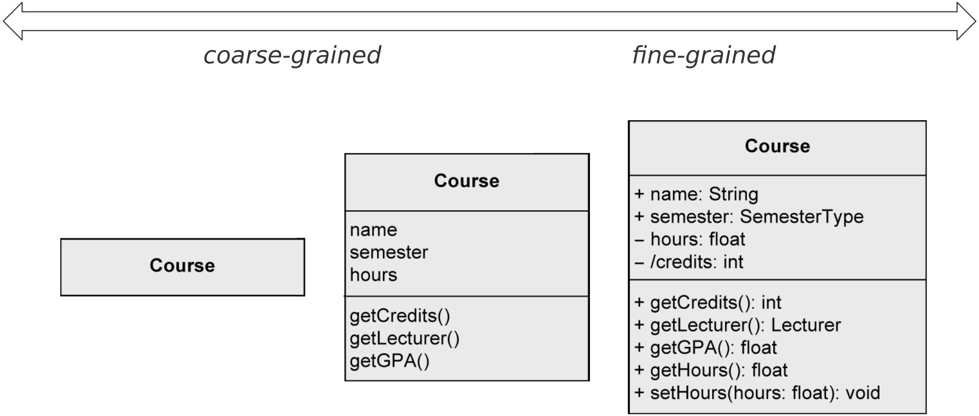
Class Variable and Class Operation

- Instance variable (= instance attribute): attributes defined on instance level
- Class Variable (= class attribute, static attribute)
  - Defined only once per class, i.e., shared by all instances of the class
  - E.g. counters for the number of instances of a class, constants, etc.
- Class operation (= static operation)
  - Can be used if no instance of the corresponding class was created
  - E.g. constructors, counting operations, math functions (sin(x)), etc.
- Notation: underlining name of class variable/class operation



```
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()  {…}
}
```

Specification of Classes



coarse-grained

fine-grained

Course

Course

name
semester
hours

getCredits()
getLecturer()
getGPA()

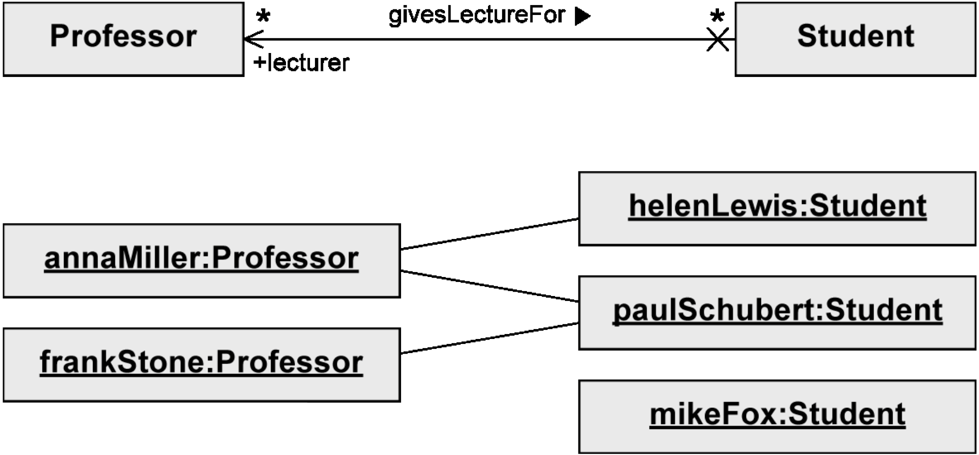
Course

+ name: String
+ semester: SemesterType
– hours: float
– /credits: int

+ getCredits(): int
+ getLecturer(): Lecturer
+ getGPA(): float
+ getHours(): float
+ setHours(hours: float): void

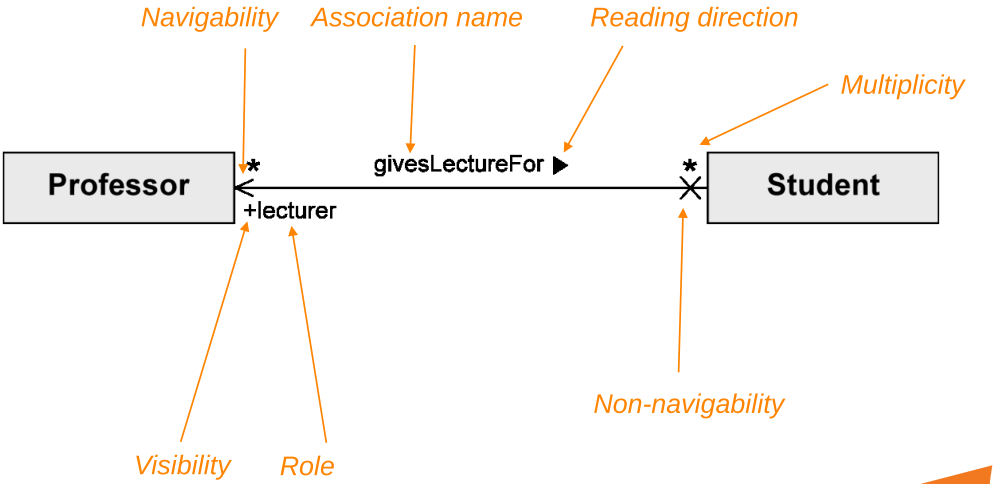
Association

- Models possible relationships between instances of classes



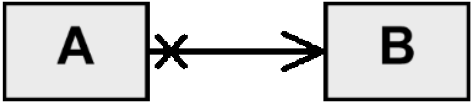
Binary Association

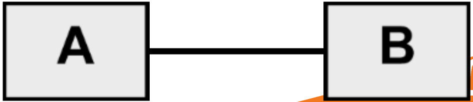
- Connects instances of two classes with one another



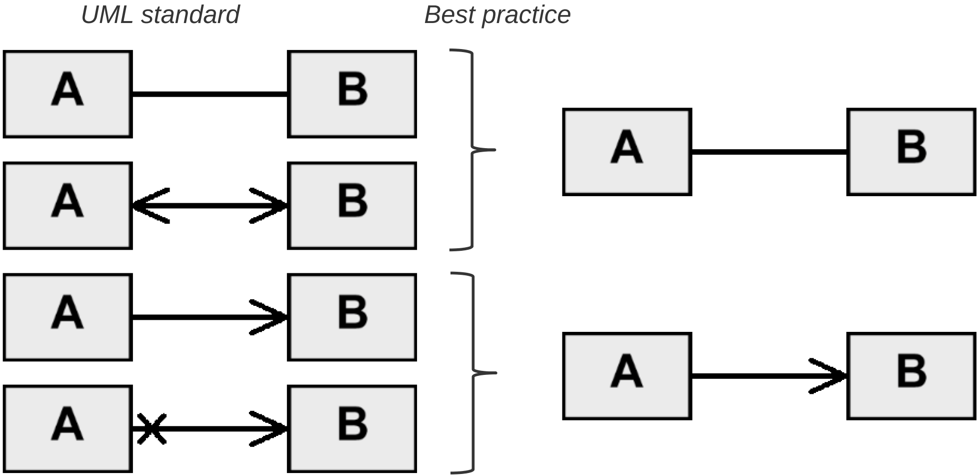
Binary Association - Navigation

- Navigability: an object knows its partner objects and can therefore access their visible attributes and operations
  - Indicated by an open arrow head
- Non-navigability
  - Indicated by cross
- Example:
  - **A** can access the visible attributes and operations of **B**
  - **B** cannot access any attributes and operations of **A**
- Navigability undefined
  - Bidirectional navigability is assumed

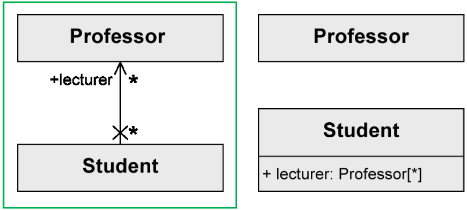




Navigability Best Practices



Binary Association as Attribute



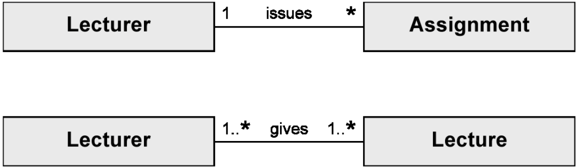
Preferable

- Java-like notation:

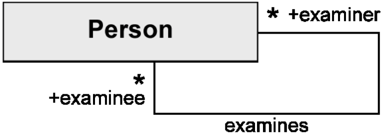
```
class Professor {…}

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

Multiplicity and Role

- Multiplicity: Number of objects that may be associated with exactly one object of the opposite side

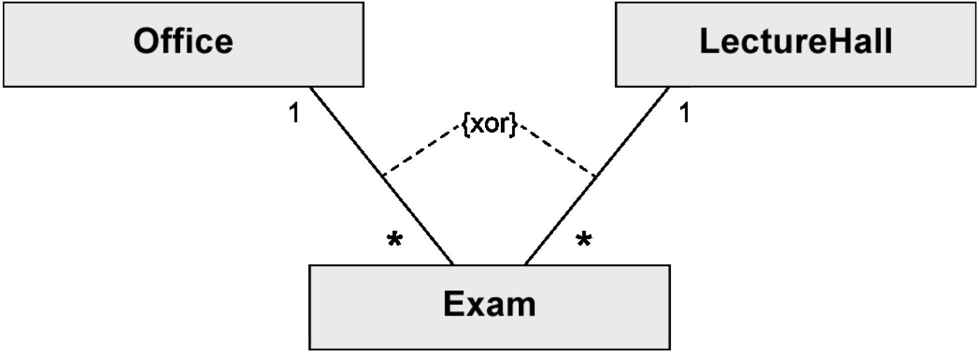


- Role: describes the way in which an object is involved in an association relationship

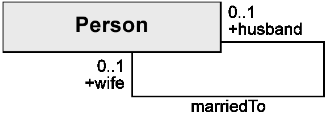


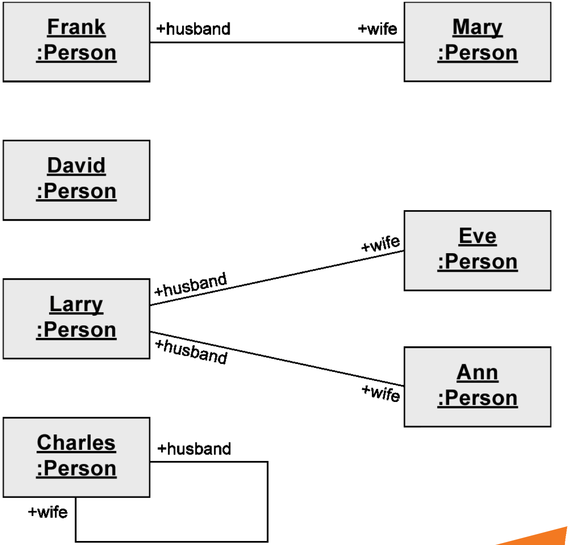
`xor` Constraint

- “exclusive or” constraint
- An object of class **A** is to be associated with an object of class **B** or an object of class **C** but not with both

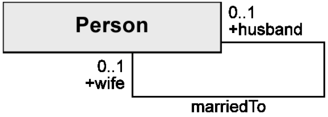


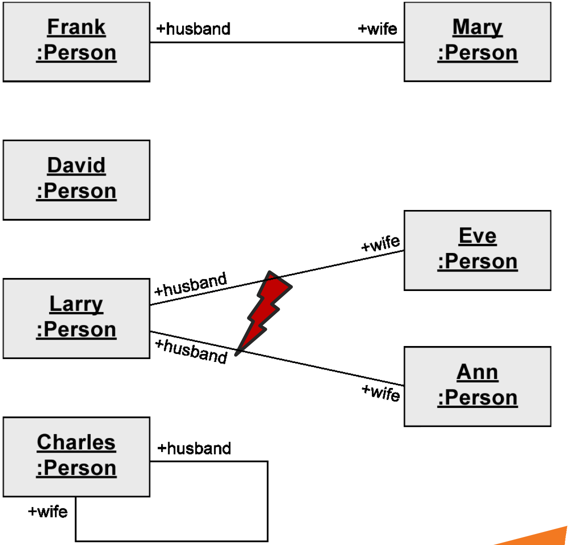
Unary Association





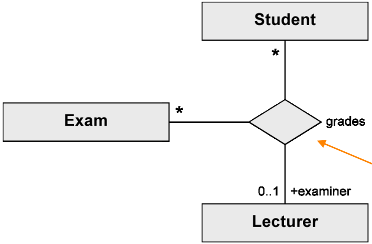
Unary Association





n-ary Associations

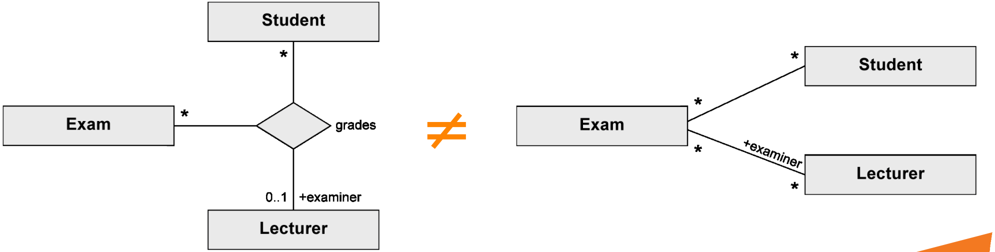
- More than two partner objects are involved in the relationship
- No navigation directions



Ternary association

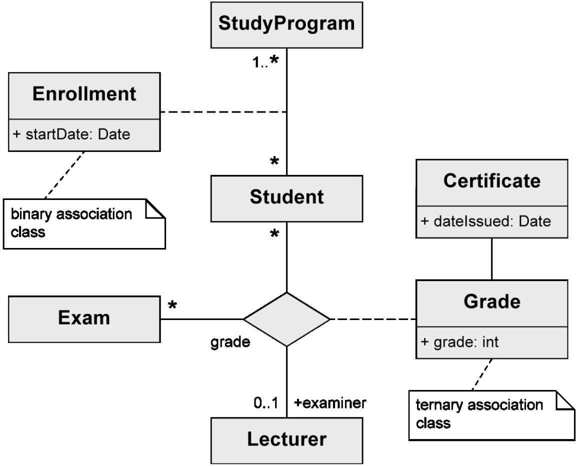
n-ary Associations

- Example
  - (Student, Exam) -> (Lecturer)
    - One student takes one exam with one or no lecturer
  - (Exam, Lecturer) -> ‘(Student)“
    - One exam with one lecturer can be taken by any number of students
  - (Student, Lecturer) -> (Exam)
    - One student can be graded by one **Lecturer** for any number of exams

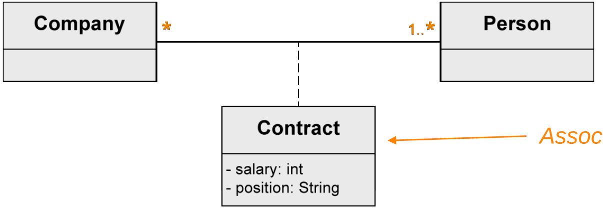


Association Class

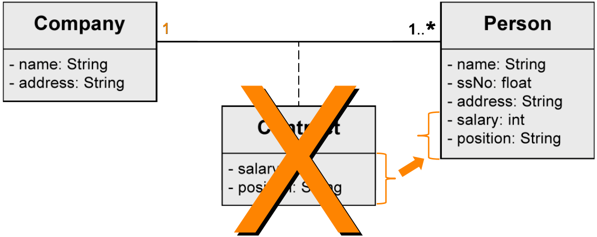
- Assign attributes to the relationship between classes rather than to a class itself



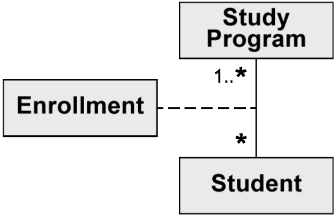
Association Class



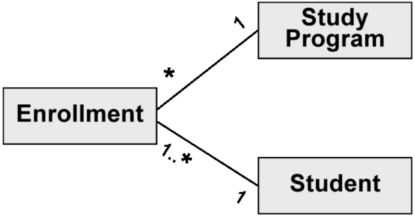
Association class



Association vs. Regular Classes



≠

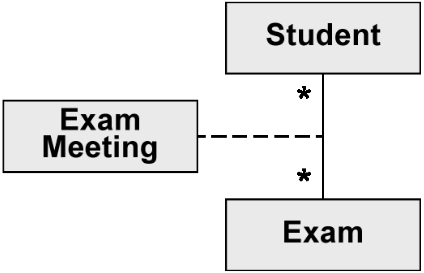


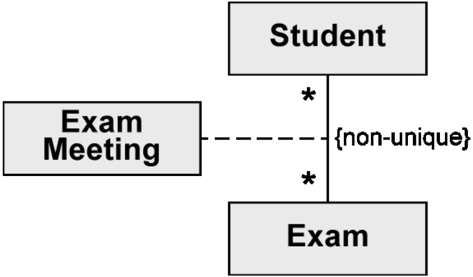
A `Student` can enroll for one particular `StudyProgram` only once

A `Student` can have mutiple `Enrollment`s for one and the same `StudyProgram`

Association Class

- Default: no duplicates
- non-unique: duplicates allowed

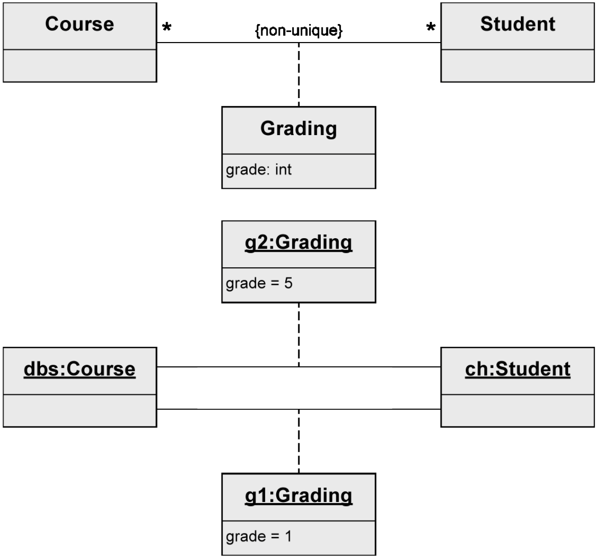


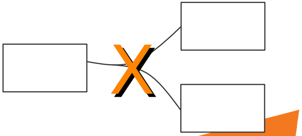


A student can only be granted an exam meeting for a specific exam once.

A student can have more than one exam meetings for a specific exam.

Association Class





Aggregation

- Special form of association
- Used to express that a class is part of another class
- Properties of the aggregation association
  - **Transitive:** if `B` is part of `A` and `C` is part of `B`, `C` is also part of `A`
  - **Asymmetric:** It is not possible for `A` to be part of `B` and `B` to be part of `A` simultaneously.
- Two types;
  - Shared aggregation
  - Composition

Are there any questions?