CPSC-224

Software Development

UML ②

Yu Wang

wangy2@gonzaga.edu

Feb 24, 2025

Announcement

- ❑ **Homework2 due day: Feb 28th**

- ❑ **Exam Day: Feb 28**

- ❑ **Final Project Preparation:**

Game Project Team Forming (Next week you need to fill in the form)

- ❑ **Homework0 due day: March 3rd**

Announcement

❑ Game Project Team Forming

- The form(Next week) is to let me know if you have anyone from the class that you want to work with on the final project.
- You're welcome to give me a team of up to 4 people and I'll do my best to accommodate it (but no promises on getting to get your desired group).
- You don't need to find a full group of four. If you only have a pair or a triplet, that's fine. I'll fit smaller pairs and triplets into teams as possible.
- Be sure to talk with the people before you try to make a team

Daily Attendance (01)

- Scan the QR Code

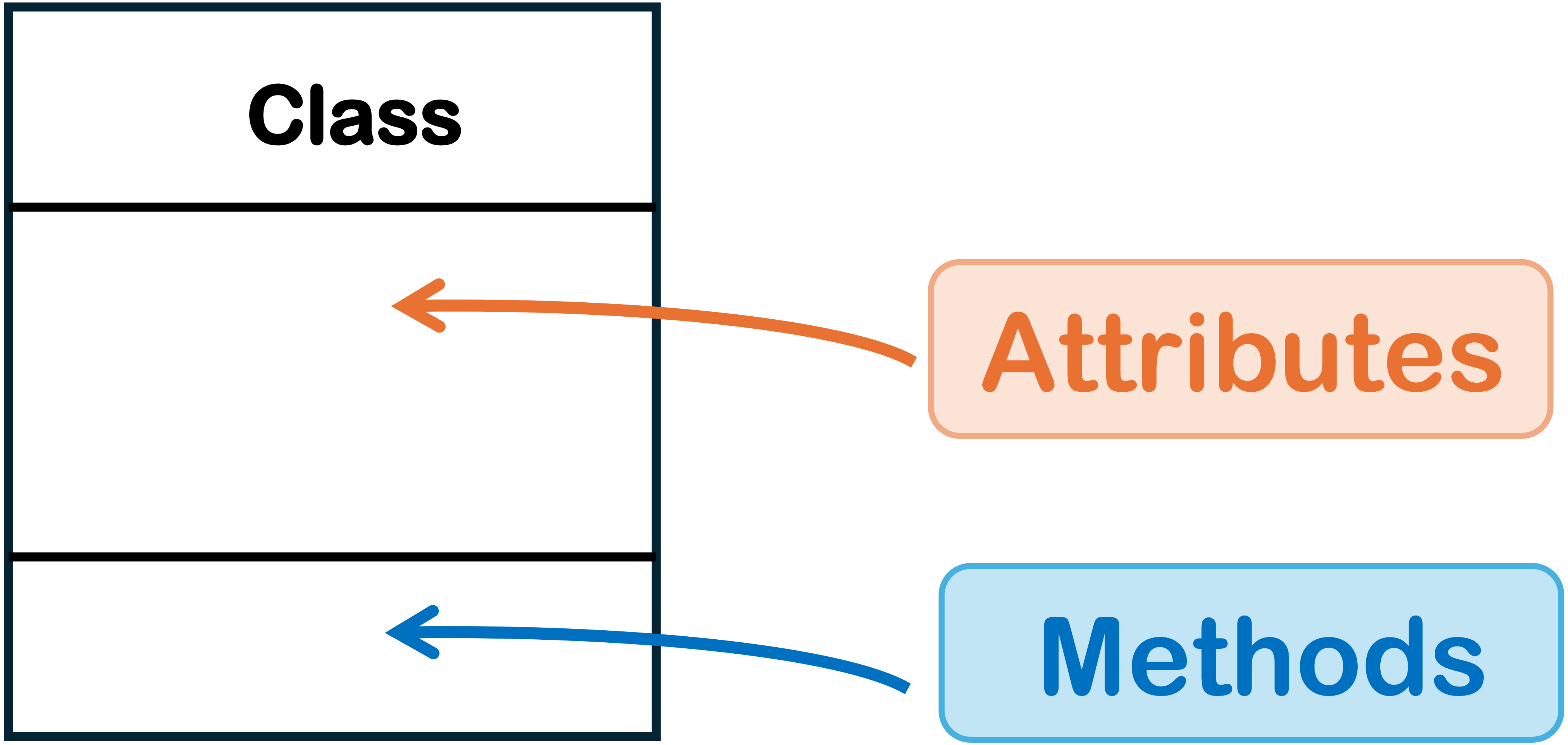
Daily Attendance (02)

- ❑ **Scan the QR Code**

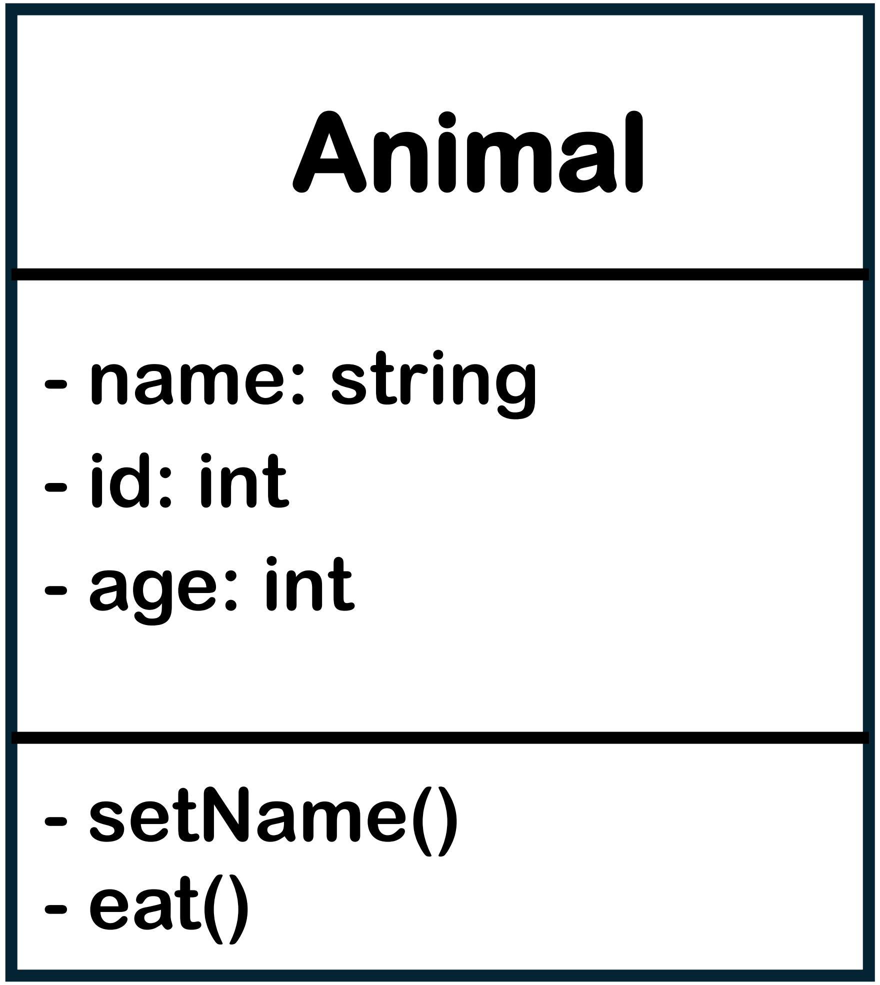
Review - Last Class

- ✓ We learned – What is UML?(UML definition)
- ✓ We learned – Why use UML?
- ✓ We learned – UML is divided into two main categories: Structural Diagrams(Static) and Behavioral Diagrams(Dynamic)
- ✓ We learned – Big picture of UML Diagrams
- ✓ We learned – Class Diagram

UML – Class Diagram Example



UML – Class Diagram Example



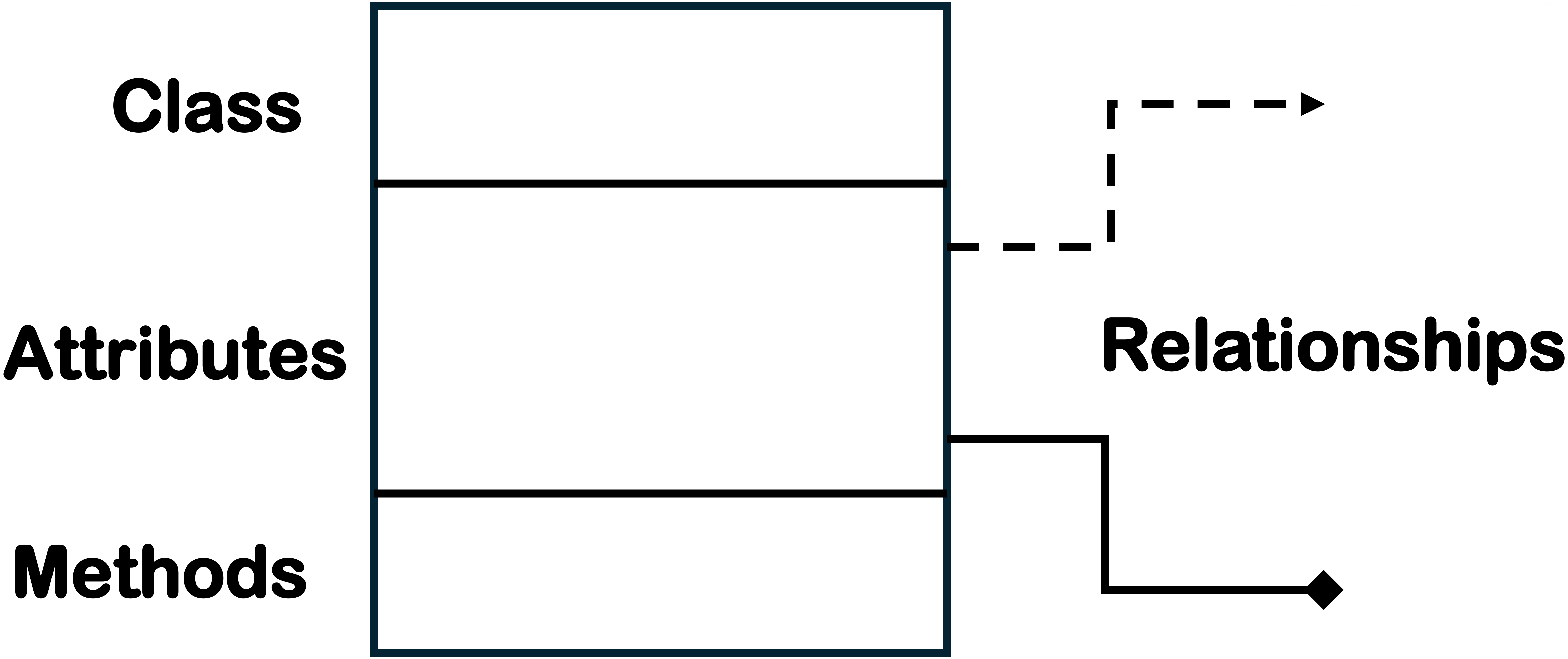
Animal

- name: string
- id: int
- age: int

- setName()
- eat()

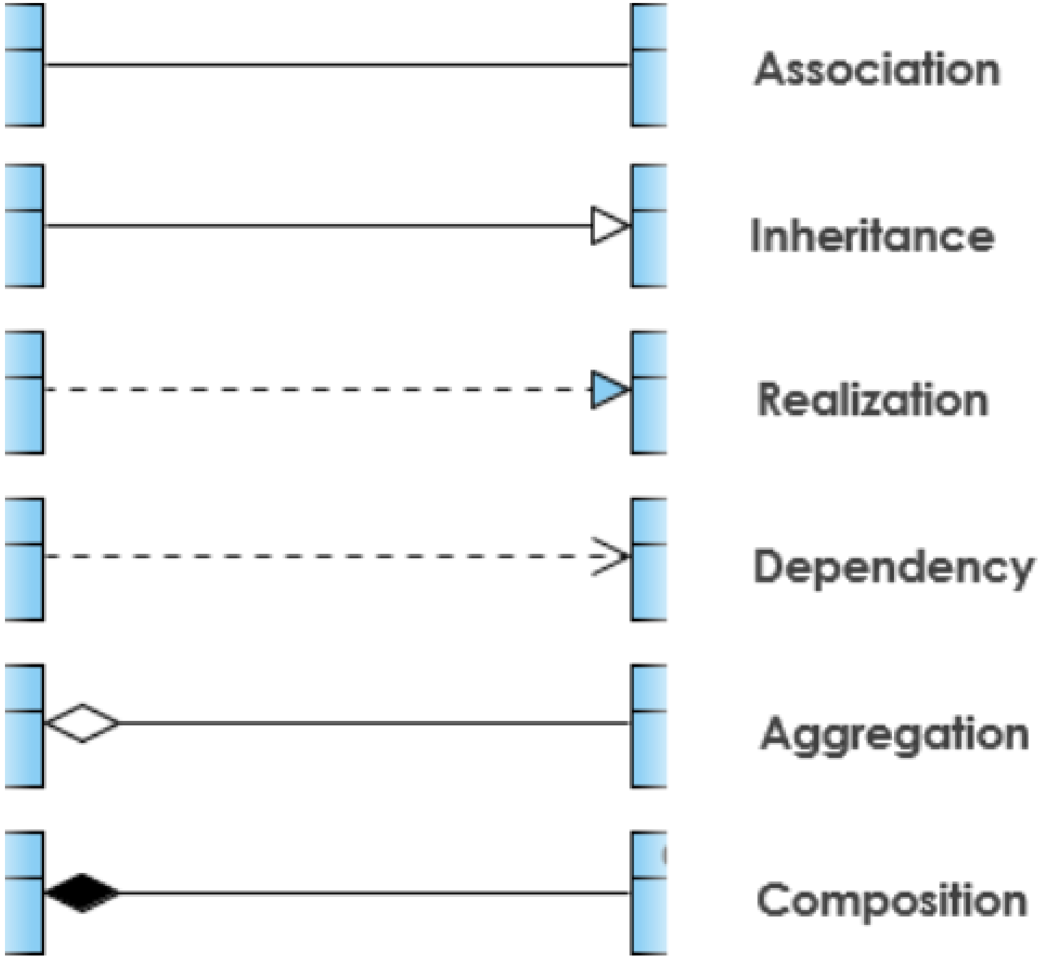
name: XiangXiang
id: 99
age: 20

UML – Class Diagram Example

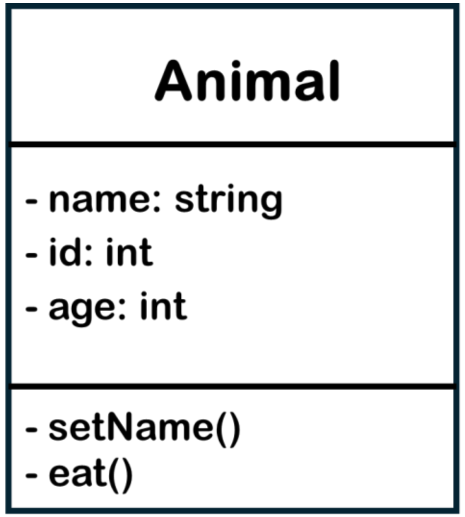


UML – Relationships between classes

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:



UML – Inheritance Relationship

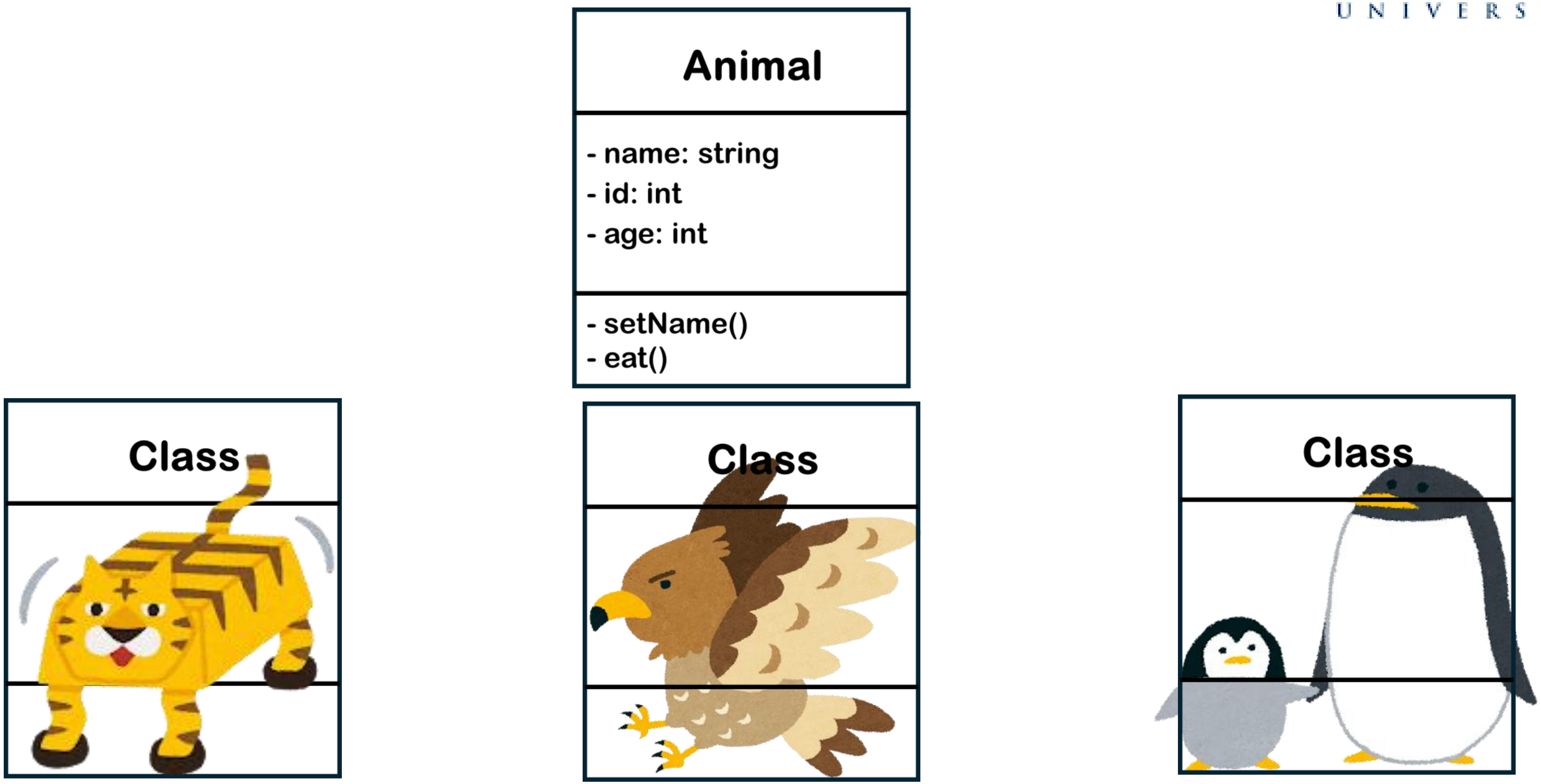


Animal

- name: string
- id: int
- age: int

- setName()
- eat()

UML – Inheritance Relationship



Animal

- name: string
- id: int
- age: int

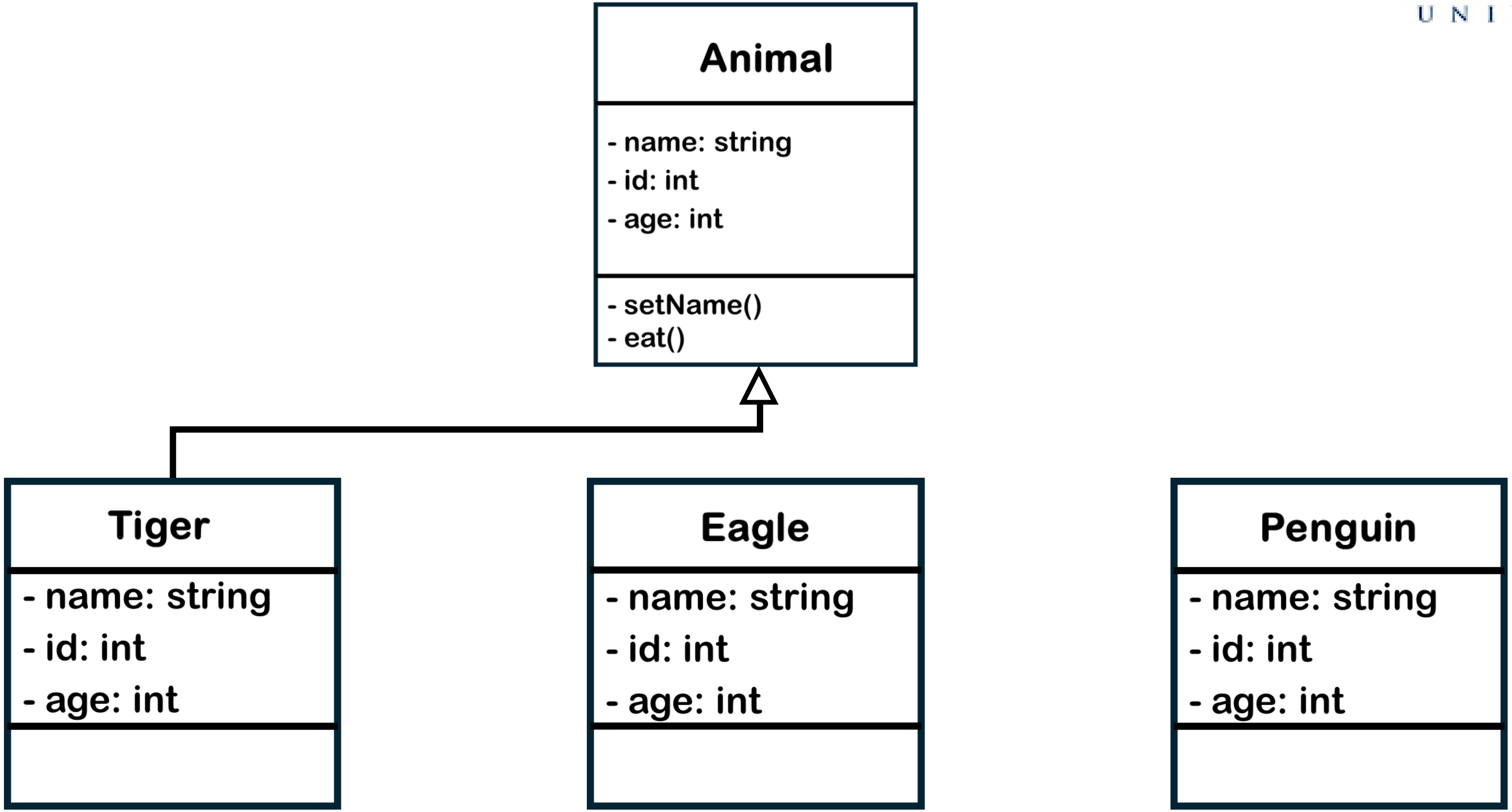
- setName()
- eat()

Class

Class

Class

UML – Inheritance Relationship



Animal

- name: string
- id: int
- age: int

- setName()
- eat()

Tiger

- name: string
- id: int
- age: int

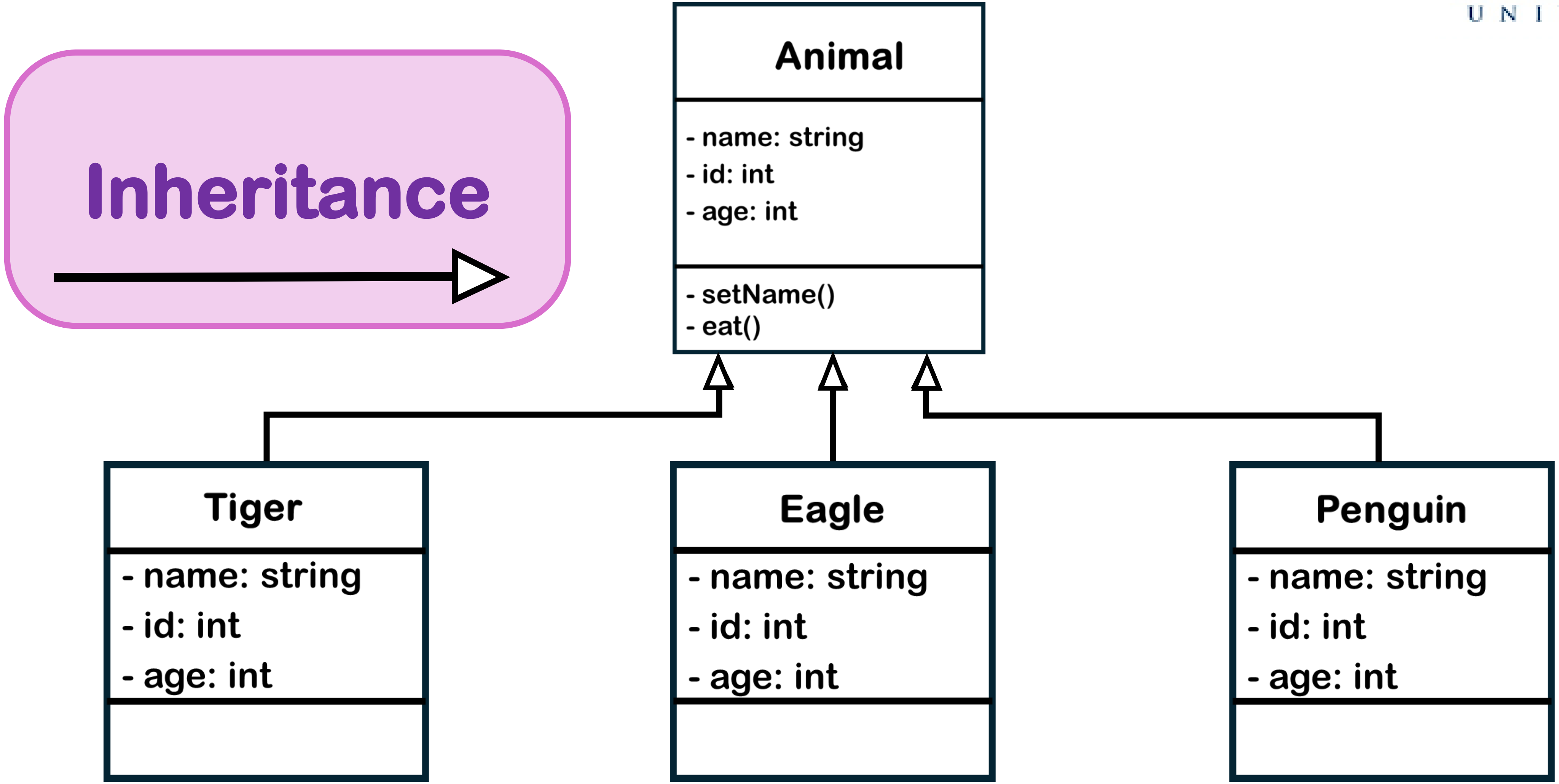
Eagle

- name: string
- id: int
- age: int

Penguin

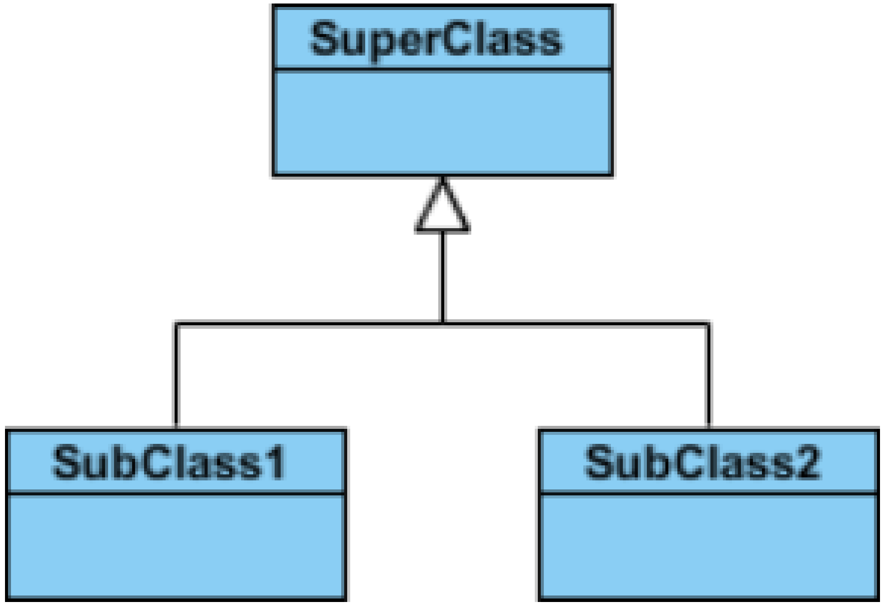
- name: string
- id: int
- age: int

UML – Inheritance Relationship

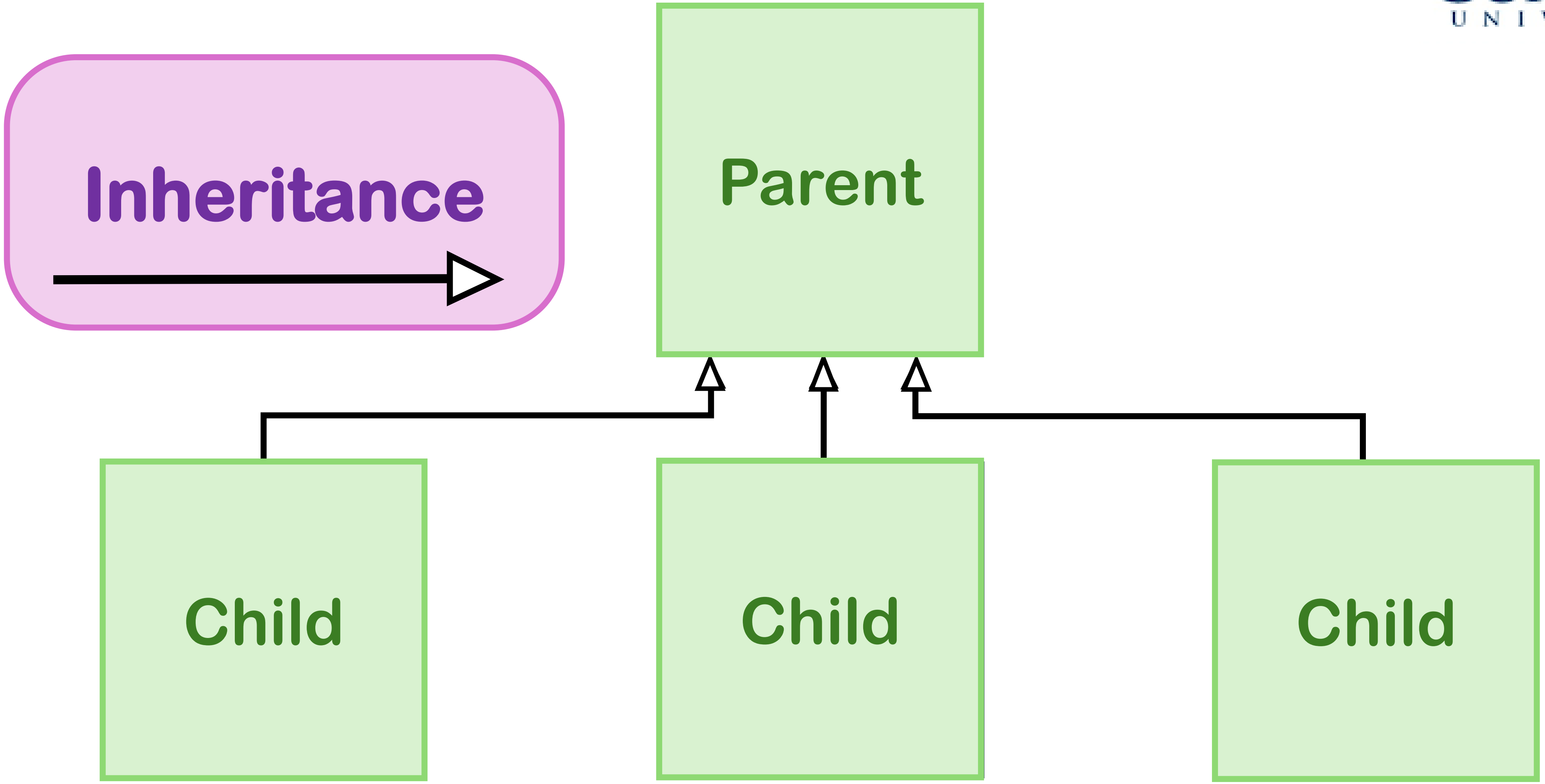


UML –Inheritance Relationship

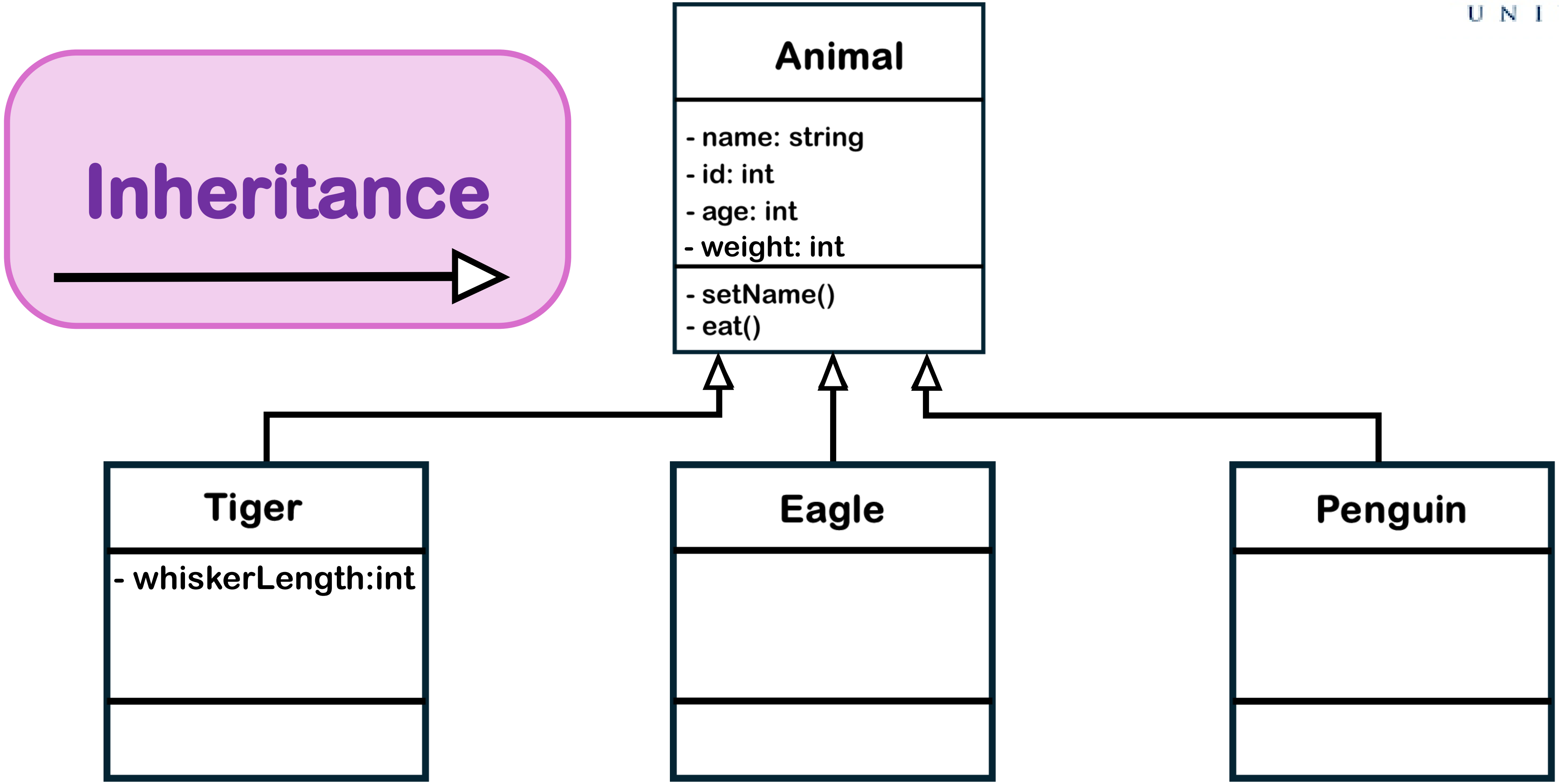
- The figure to the right shows an example of inheritance hierarchy. SubClass1 and SubClass2 are derived from SuperClass.
- The relationship is displayed as a solid line with a hollow arrowhead that points from the child element to the parent element.



UML – Inheritance Relationship



UML – Inheritance Relationship



Inheritance

Animal

- name: string
- id: int
- age: int
- weight: int

- setName()
- eat()

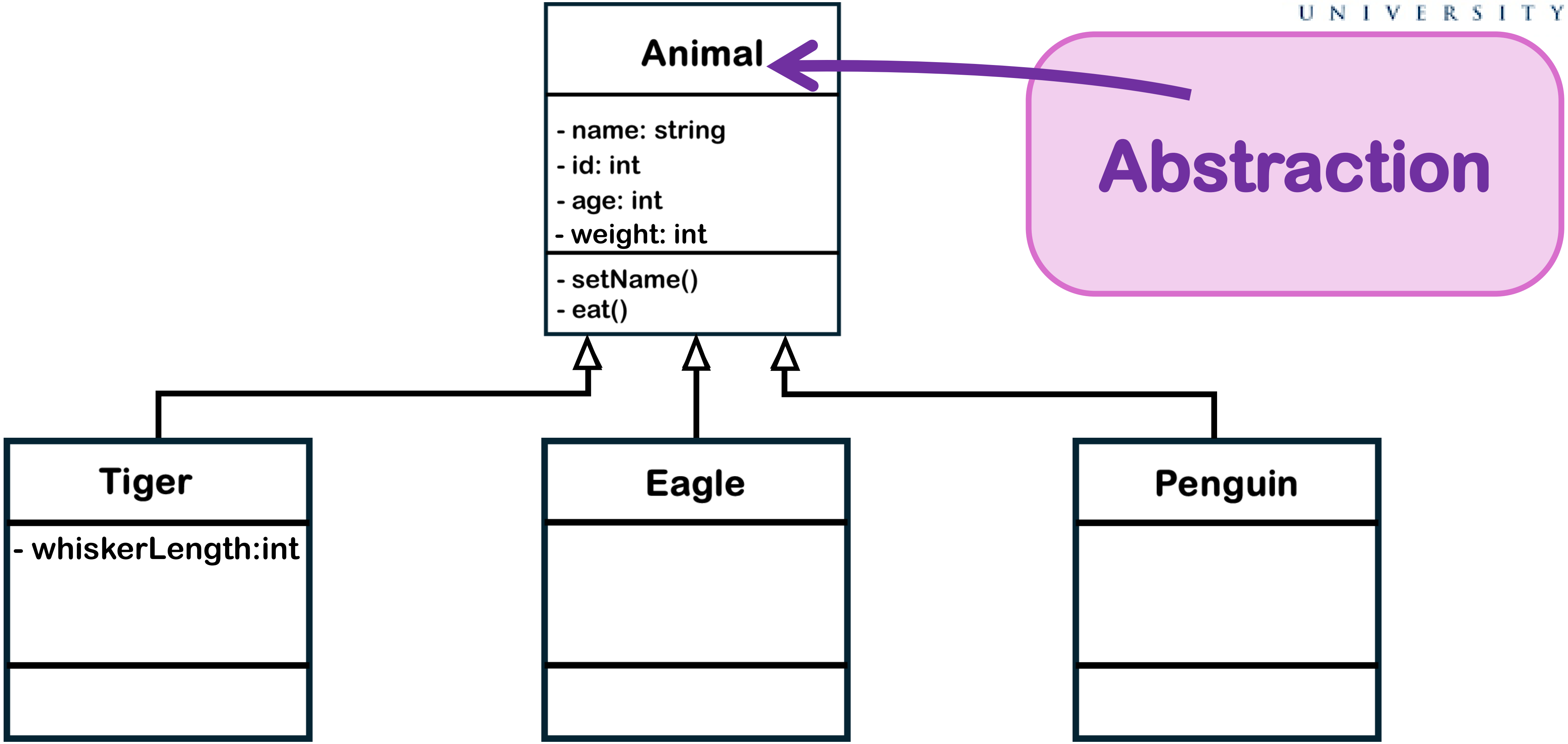
Tiger

- whiskerLength:int

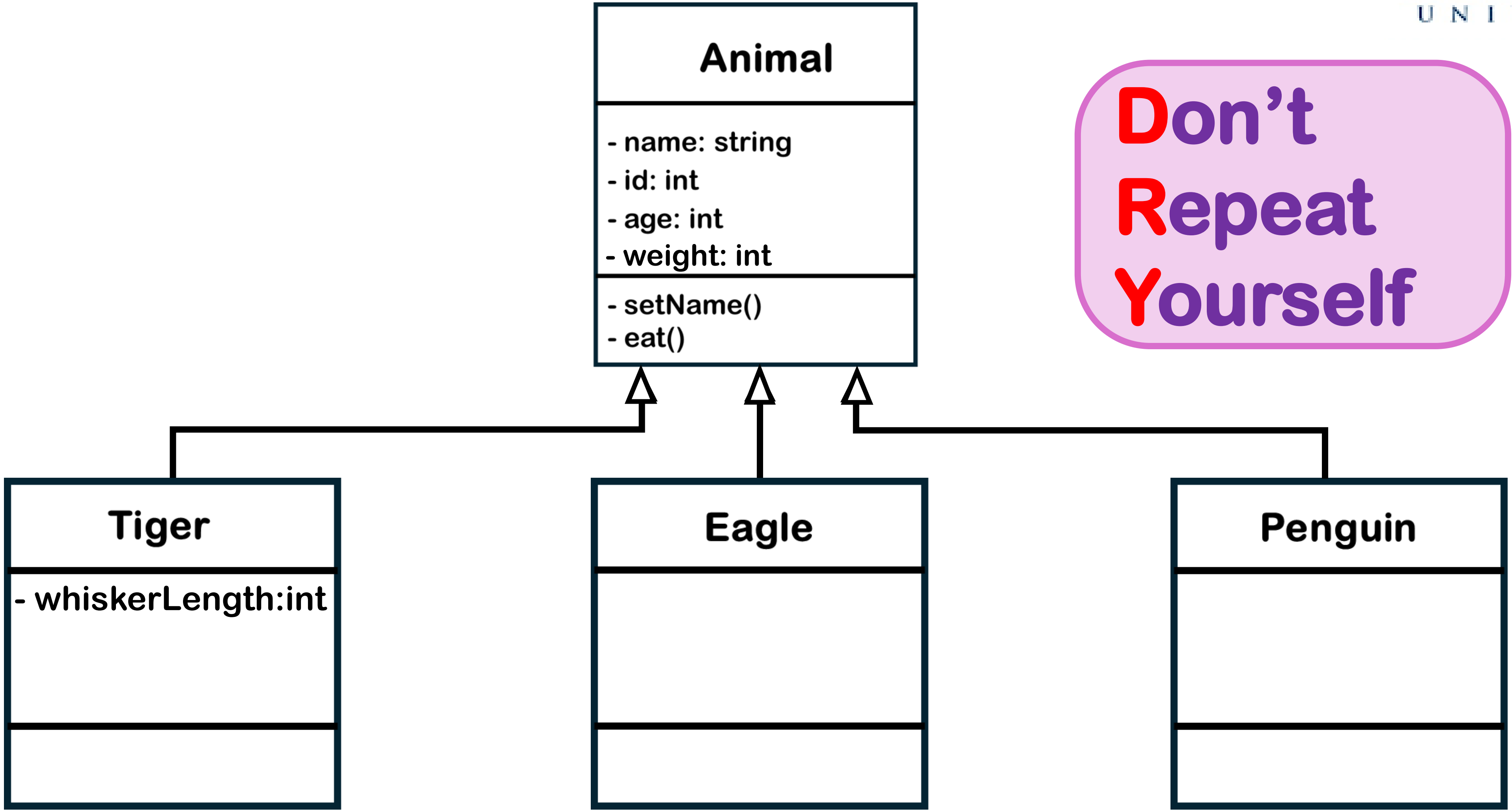
Eagle

Penguin

UML – Inheritance Relationship

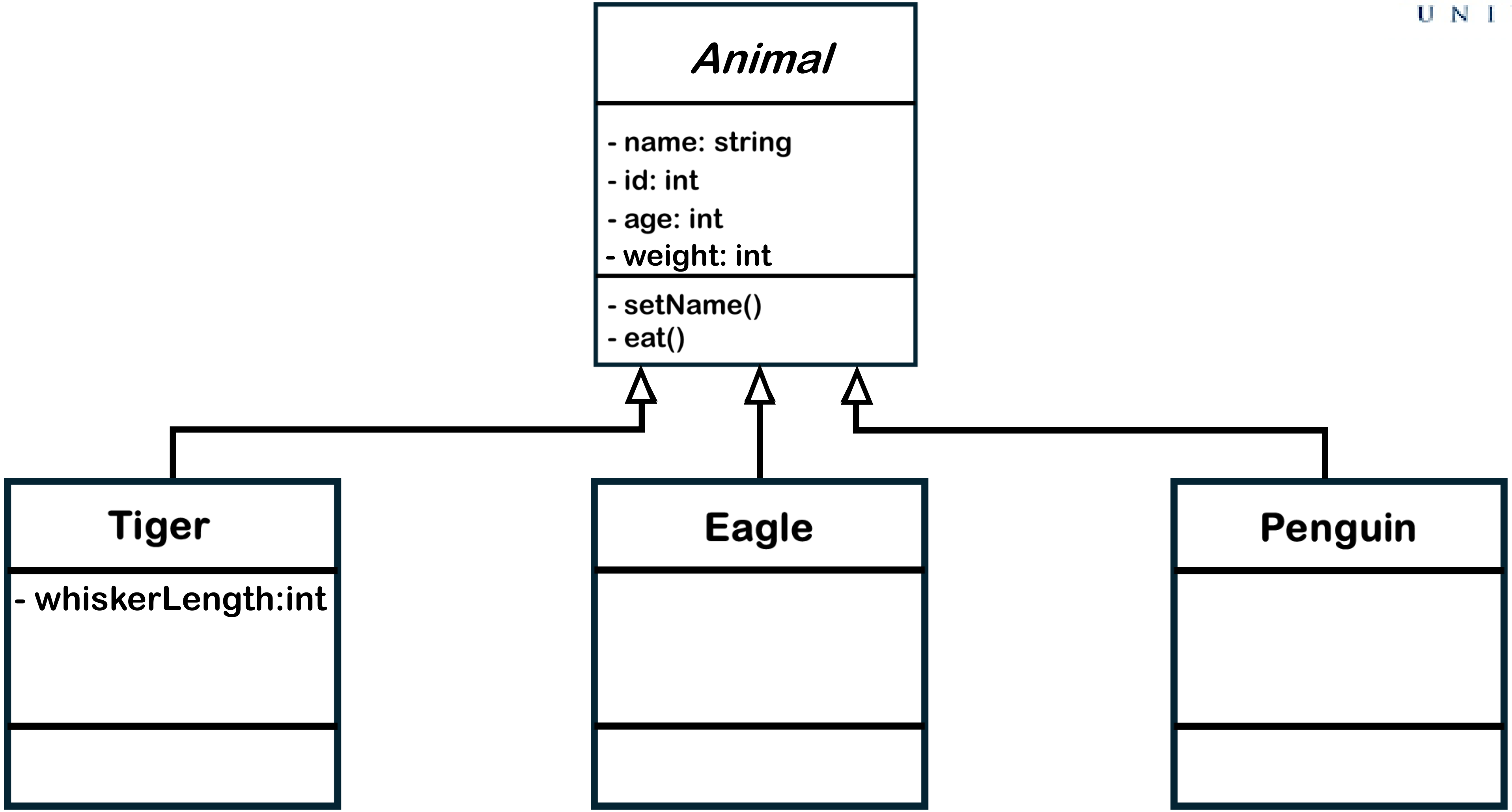


UML – Inheritance Relationship

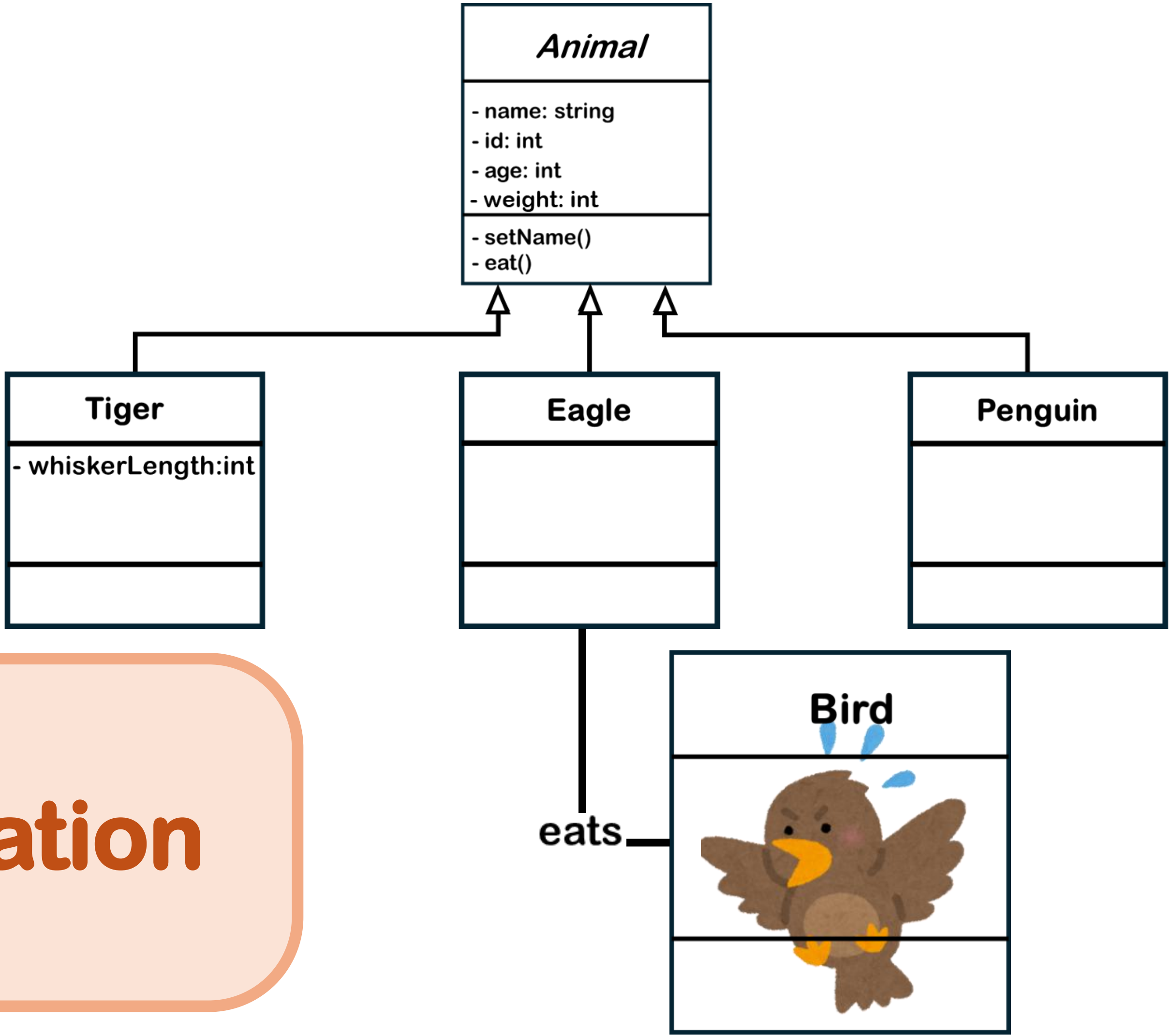


Don't
Repeat
Yourself

UML – Abstract Class

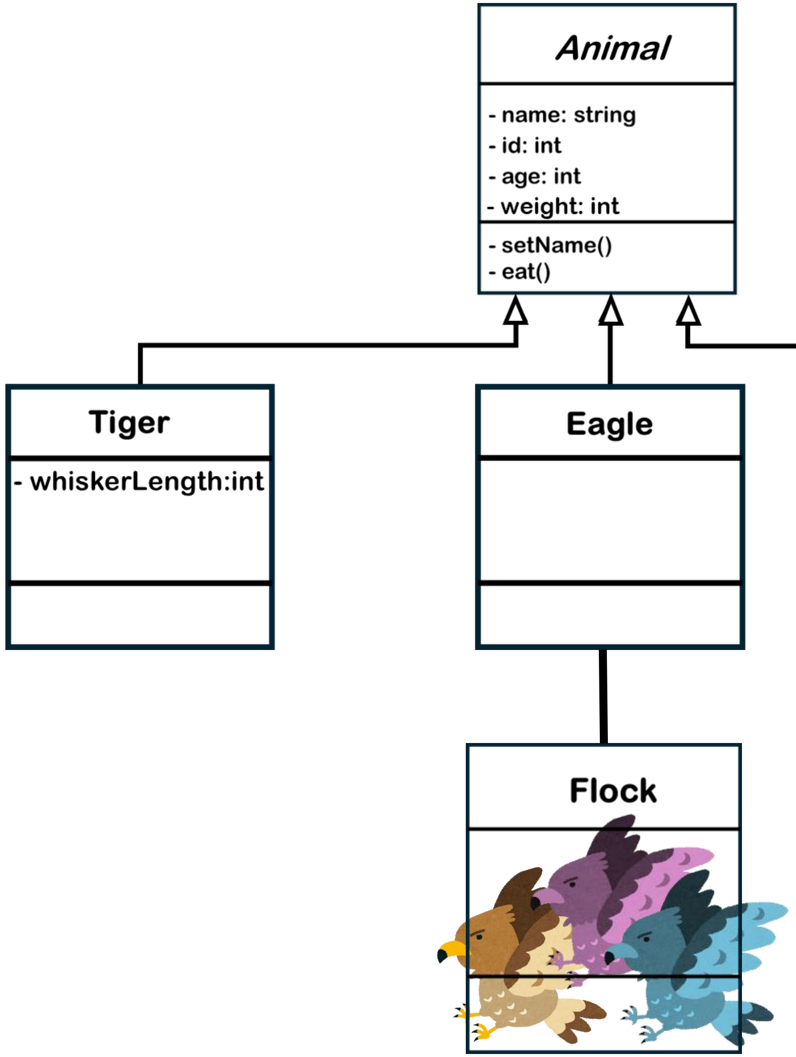


UML – Association Relationship



Association

UML – Aggregation Relationship



Animal

- name: string
- id: int
- age: int
- weight: int

- setName()
- eat()

Tiger

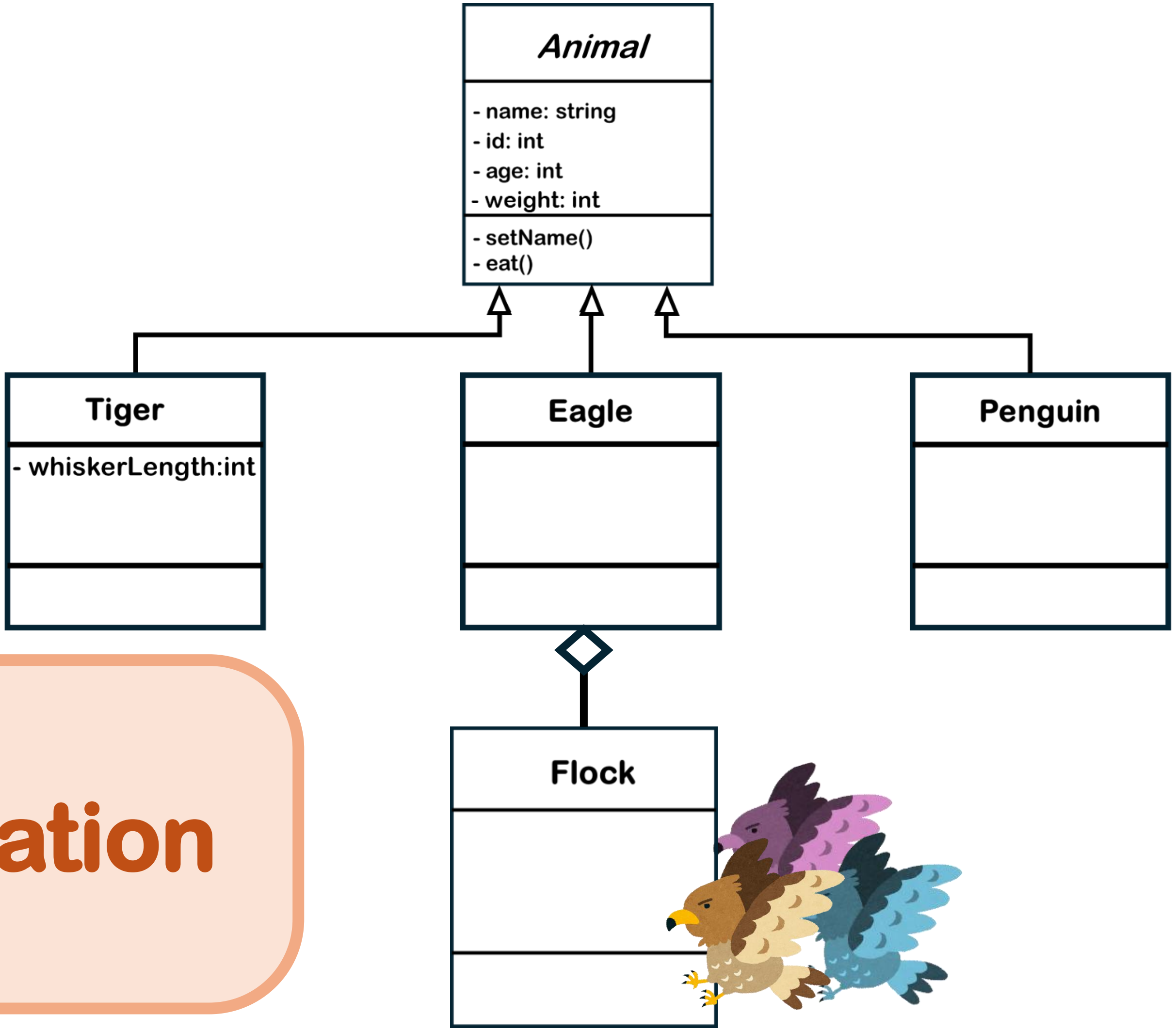
- whiskerLength:int

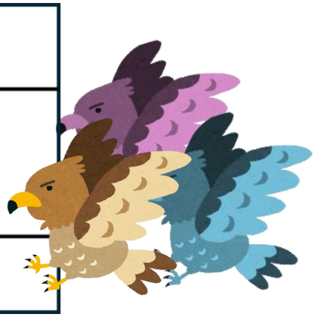
Eagle

Penguin

Flock

UML – Aggregation Relationship

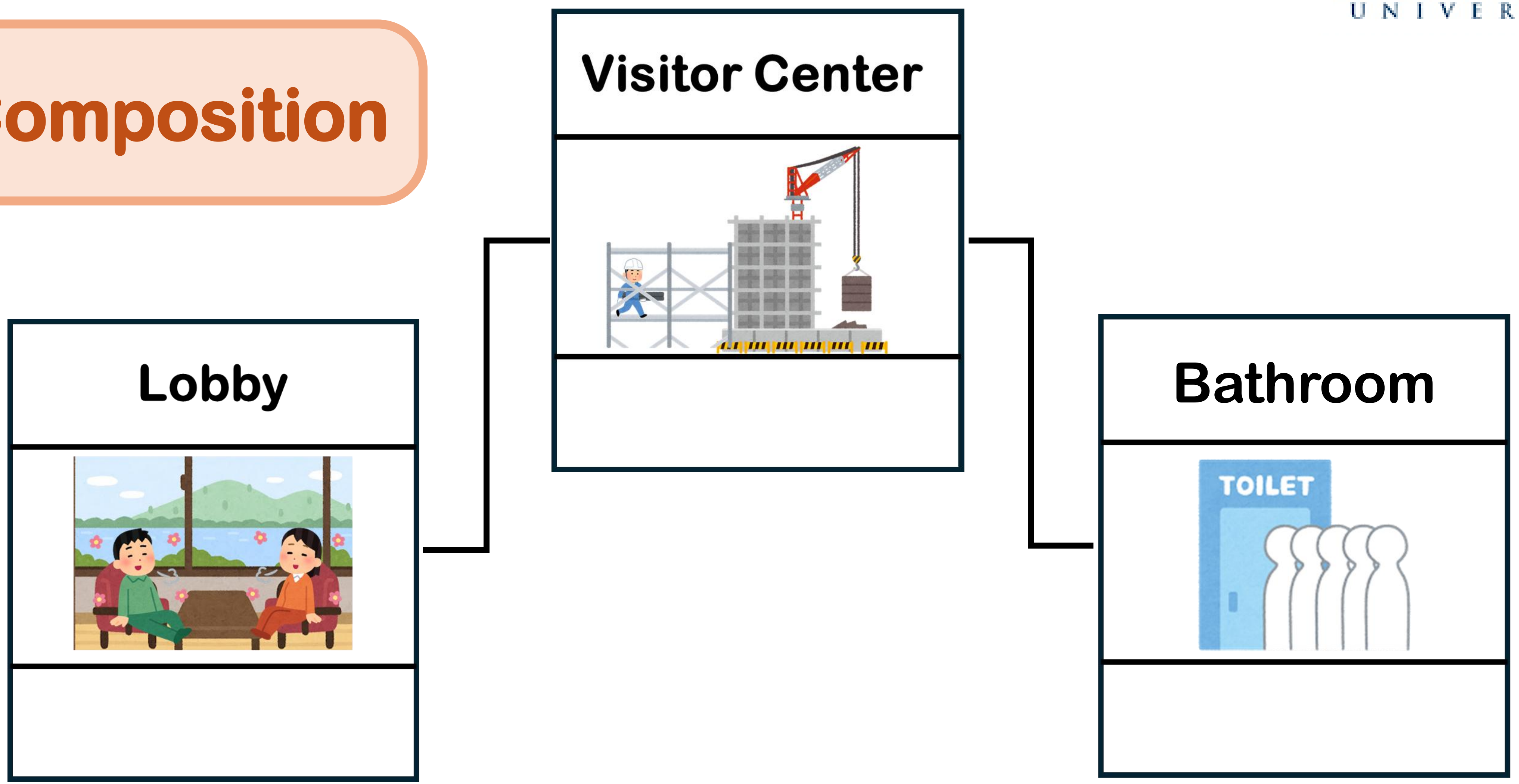




Aggregation

UML – Composition Relationship

Composition

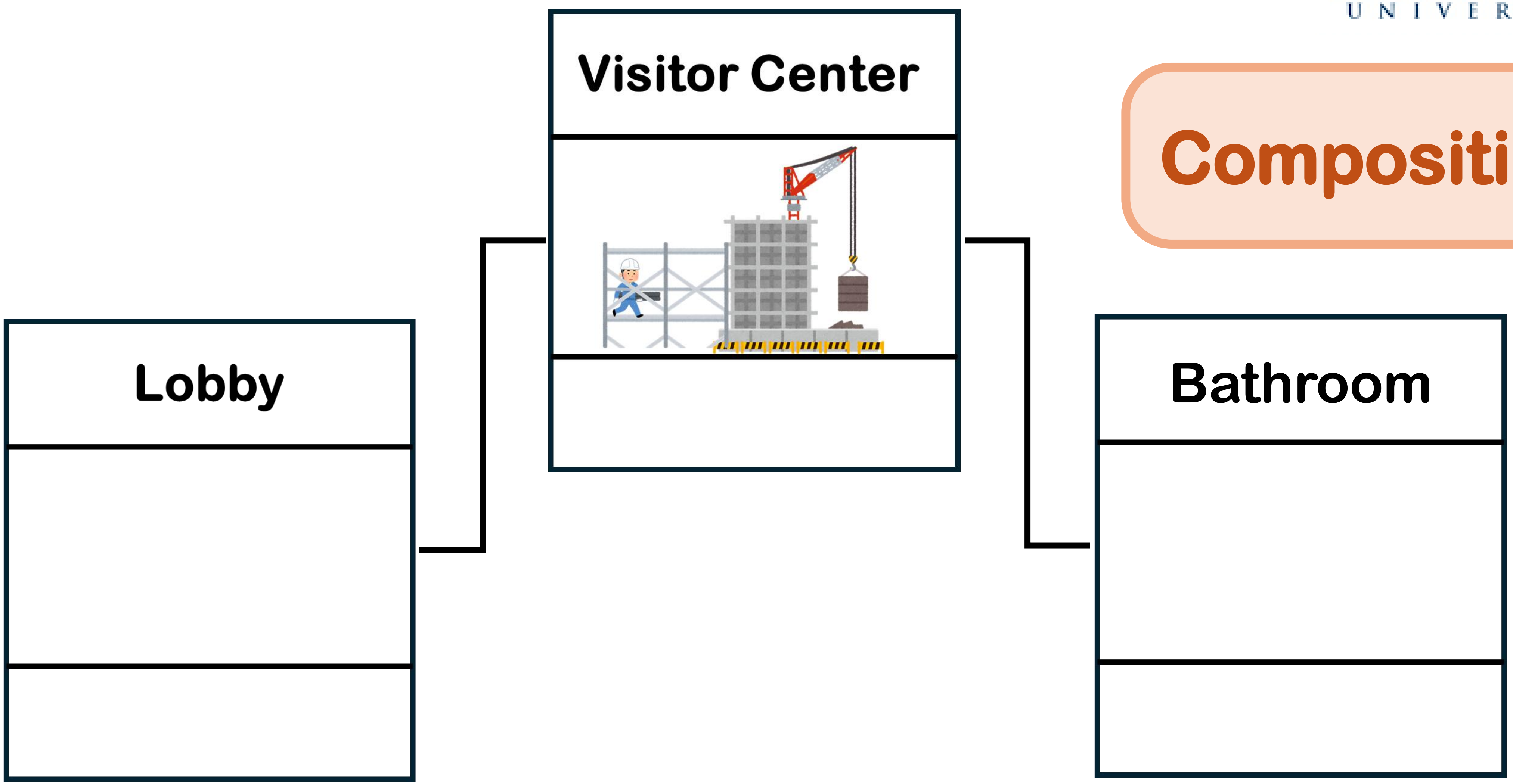


Visitor Center

Lobby

Bathroom

UML – Composition Relationship



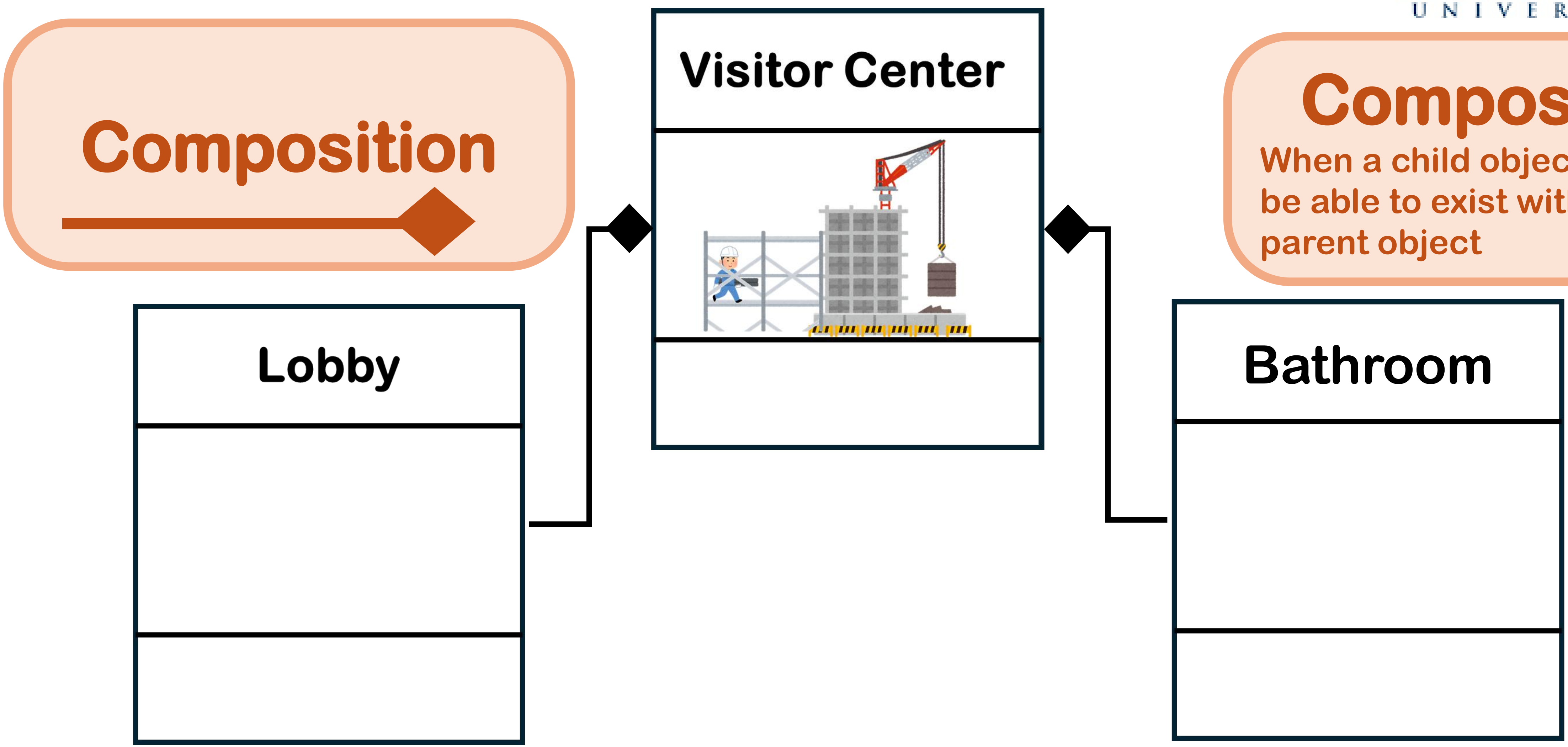
Visitor Center

Lobby

Bathroom

Composition

UML – Composition Relationship



Composition

When a child object wouldn’t be able to exist without its parent object

UML – Cardinality

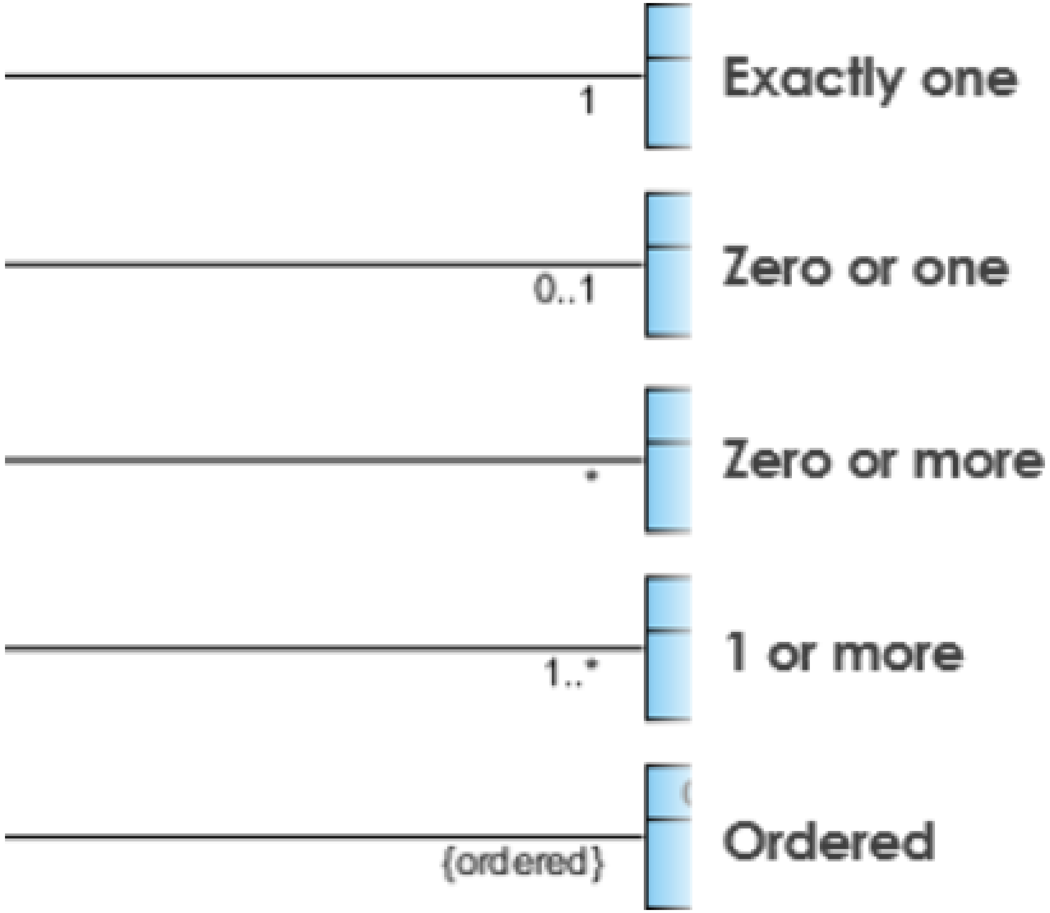
- ❖ In UML, cardinality (also called multiplicity) defines the number of instances of one class that can be associated with a single instance of another class in a relationship.

- ❖ It specifies how many objects of a given class can be related to a single object of another class.

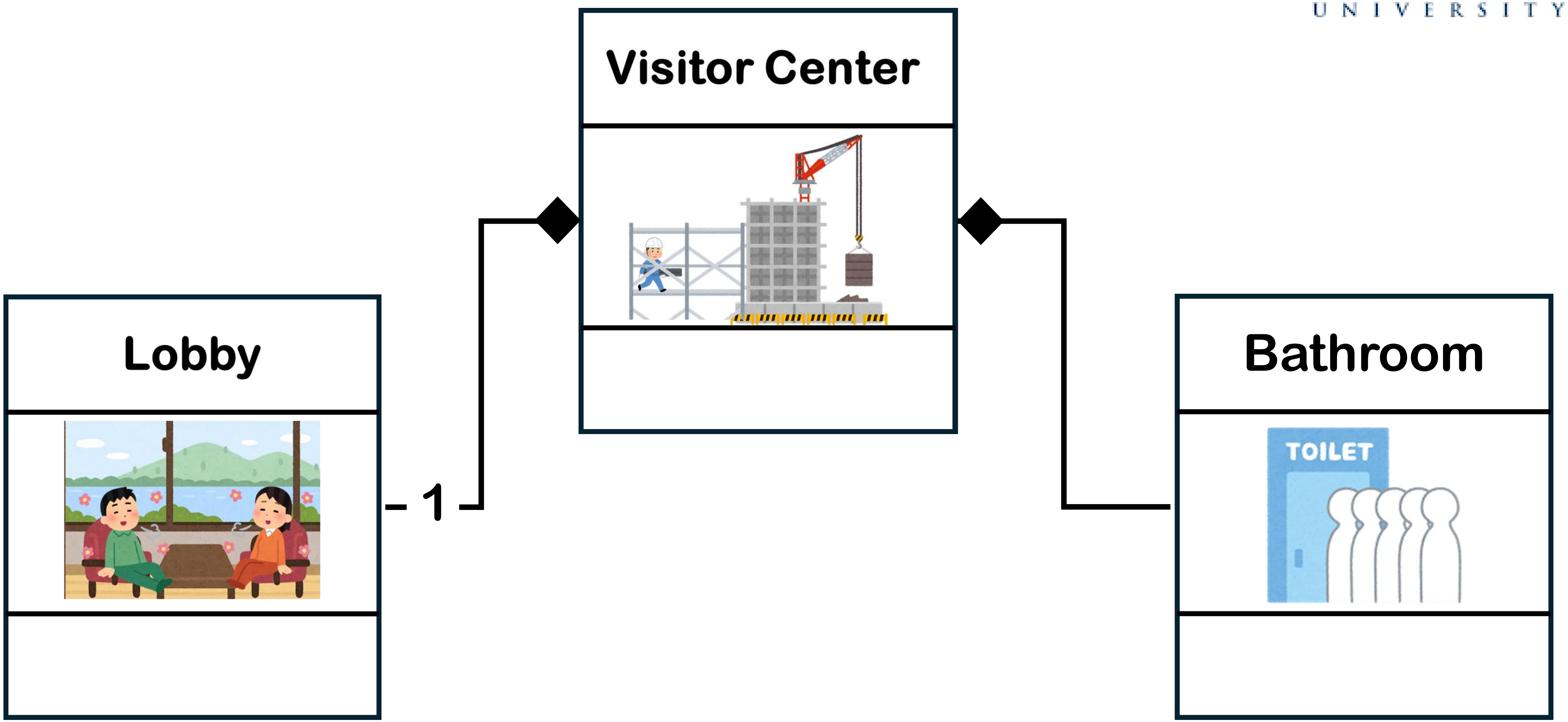
UML – Cardinality

Cardinality is expressed in terms of:

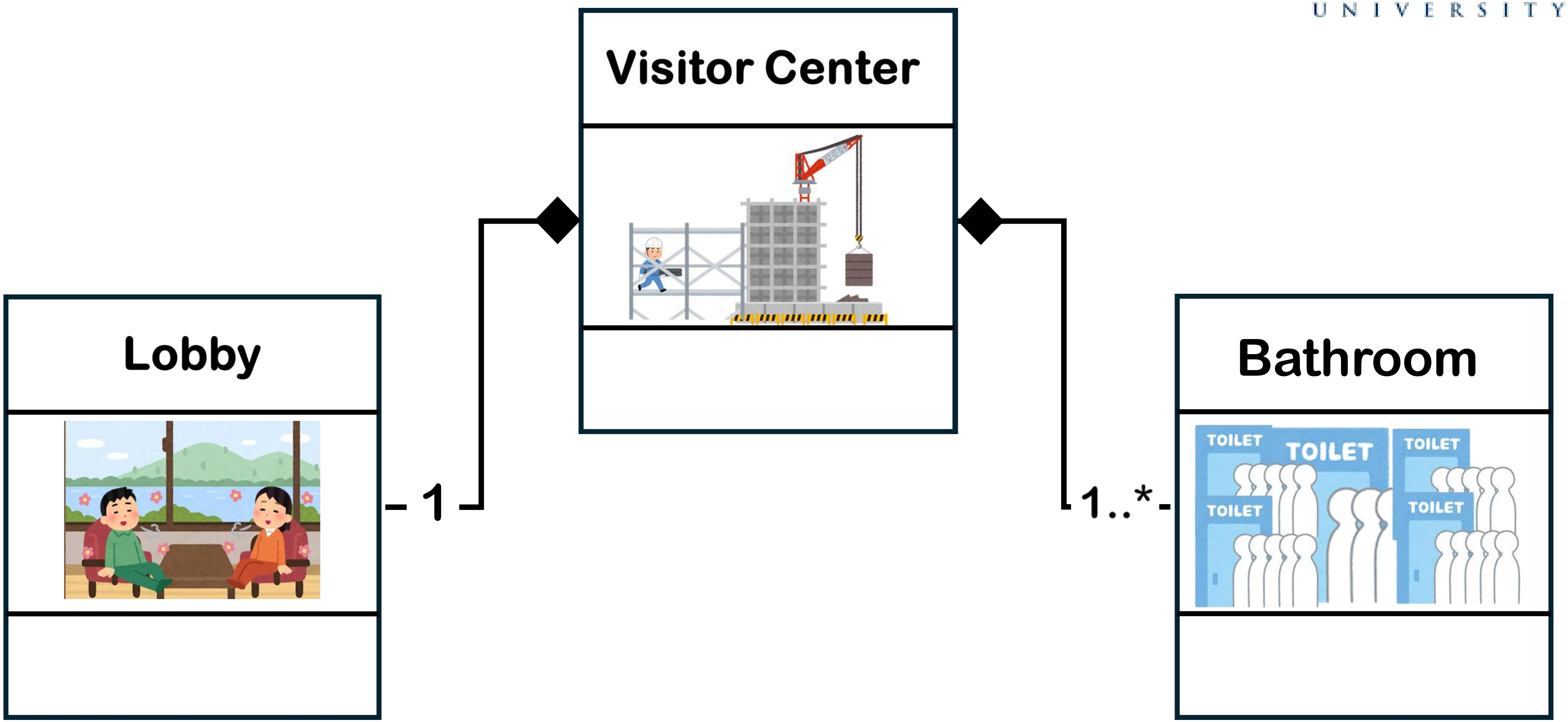
- one to one
- one to many
- many to many



UML – Composition Relationship



UML – Composition Relationship



Visitor Center

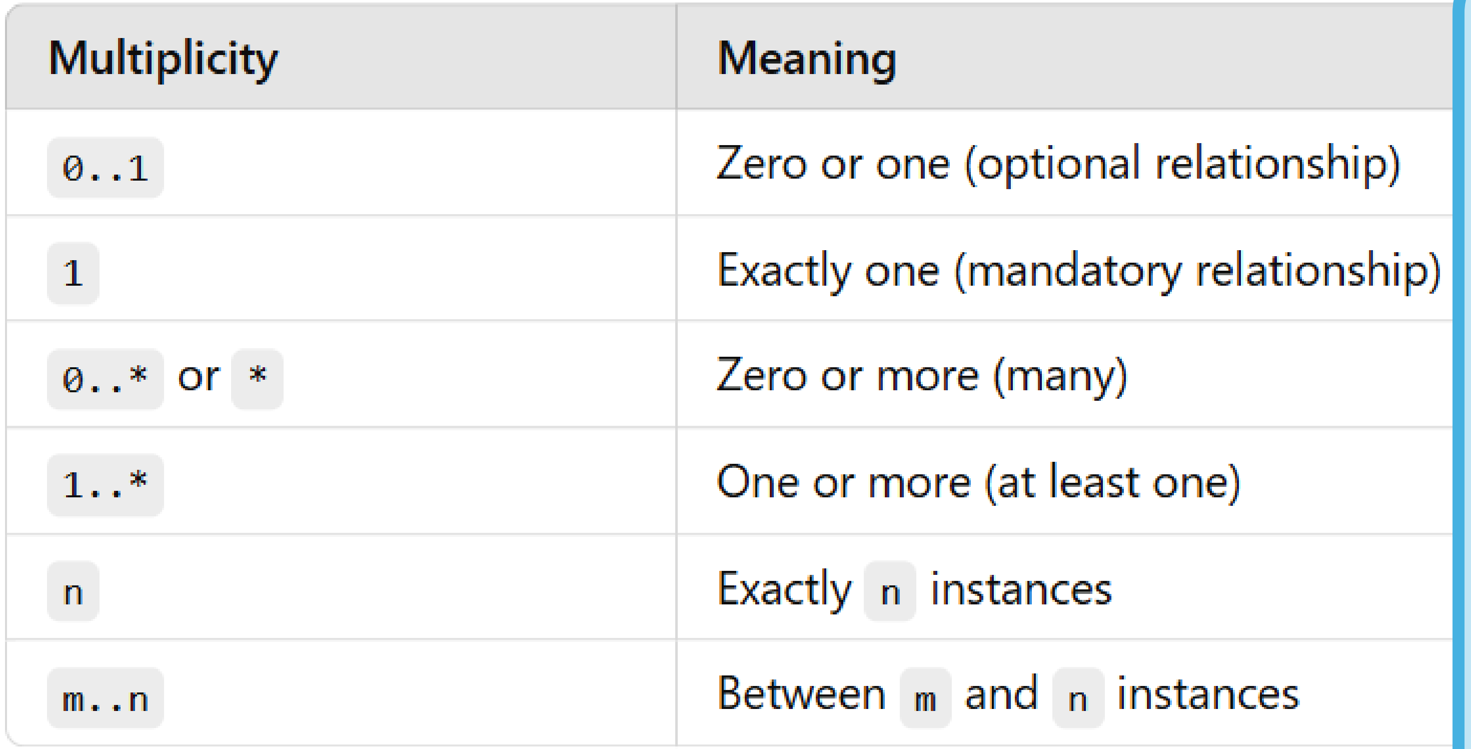
Lobby

1

Bathroom

1..*

UML – Cardinality Notations Table



Multiplicity	Meaning
`0..1`	Zero or one (optional relationship)
`1`	Exactly one (mandatory relationship)
`0..*` or `*`	Zero or more (many)
`1..*`	One or more (at least one)
`n`	Exactly `n` instances
`m..n`	Between `m` and `n` instances

- one to one (1)

Example: A Person has one Heart, and a Heart belongs to one Person.

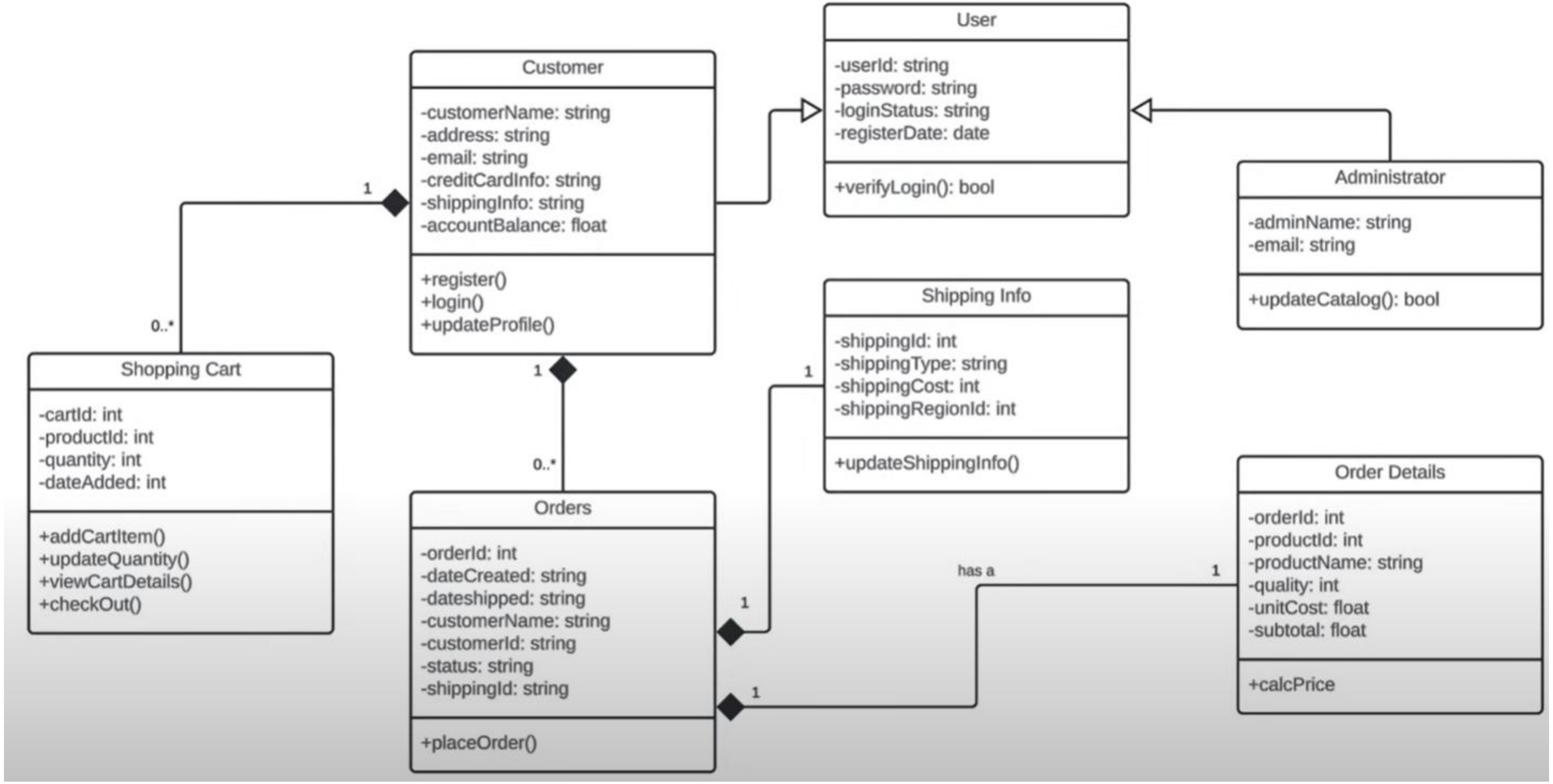
- one to many (1..* or 1..n)

Example: A Department has many Employees, but an Employee belongs to only one Department.

- many to many (*..*)

Example: A Student can enroll in multiple Courses, and a Course can have multiple Students.

Class Diagram Example: Order System



User

-userId: string
-password: string
-loginStatus: string
-registerDate: date

+verifyLogin(): bool

Customer

-customerName: string
-address: string
-email: string
-creditCardInfo: string
-shippingInfo: string
-accountBalance: float

+register()
+login()
+updateProfile()

Administrator

-adminName: string
-email: string

+updateCatalog(): bool

Shopping Cart

-cartId: int
-productId: int
-quantity: int
-dateAdded: int

+addCartItem()
+updateQuantity()
+viewCartDetails()
+checkOut()

Shipping Info

-shippingId: int
-shippingType: string
-shippingCost: int
-shippingRegionId: int

+updateShippingInfo()

Orders

-orderId: int
-dateCreated: string
-dateshipped: string
-customerName: string
-customerId: string
-status: string
-shippingId: string

+placeOrder()

Order Details

-orderId: int
-productId: int
-productName: string
-quality: int
-unitCost: float
-subtotal: float

+calcPrice

Small Group Exercise: UML Diagram Drawing

- Get into groups of 2-4
- Draw a UML Class Diagram for my project: a Hotel Reservation and Management System
- It should include things like:
  - ○ Hotel
  - ○ Room
  - ○ Guest
  - ○ Guest Loyalty levels
  - ○ Reservation - Many possible states - [future, current, past]
  - ○ Payment information
  - ○ Special room types: SuiteRoom, StandardRoom, TwinRoom, AccessibleRoom
    - ○ Some rooms have balconies
  - ○ Staff information - Maintenance, Management, DeskWorker, Cleaning Crew

Any Questions?