Using Interfaces

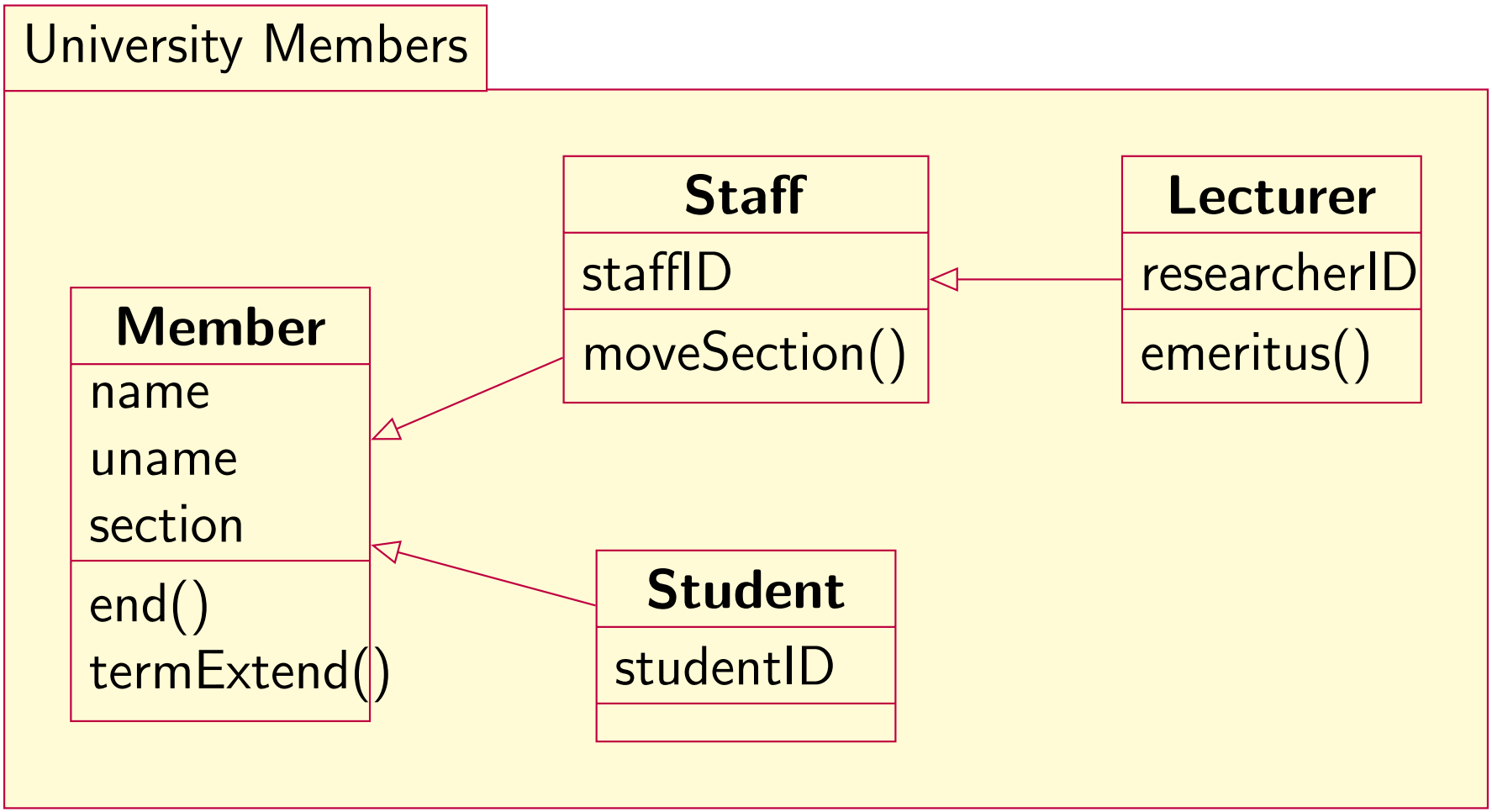
Object Oriented Programming
2022 First Semester
Shin-chi Tadaki (Saga University)

1. Hierarchical structure of classes
2. Superclasses and subclasses in Java
3. Interfaces

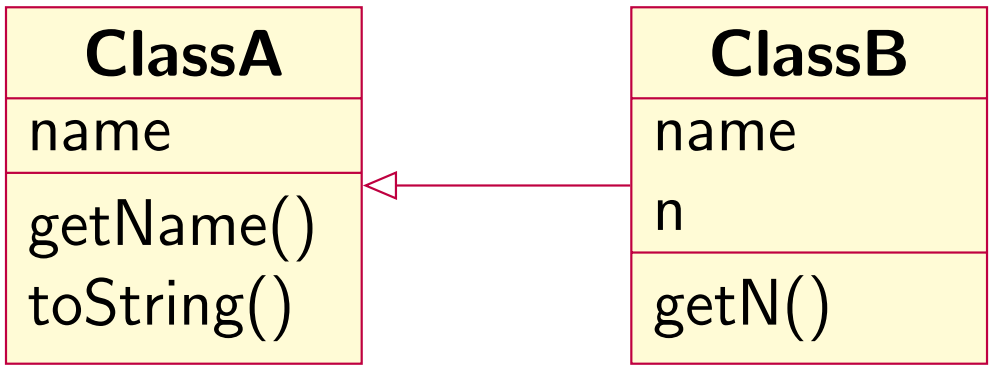
Hierarchical structure of classes: クラス階層

- Superclasses
  - Generalization / Abstraction : 一般化 / 抽象化
- Subclasses
  - embodiment / Specialization : 実装 / 具体化
- Need appropriate hierarchical structure

Example


University Members
Staff
staffID
moveSection()
Lecturer
researcherID
emeritus()
Member
name
uname
section
end()
termExtend()
Student
studentID

Simple Example



sample code
`https://github.com/oop-mc-saga/JavaIntroduction`

Inheritance: define subclass: 継承

- Inherit all fields and methods
- Additional fields and methods
- Change implementation

Generalization: define superclass: 一般化

- Common fields and methods
- Abstract methods without implementation

Method Override: メソッドの再定義

- Identifier of methods in Java: contact /signature
  - method name
  - argument list
- Implementation of abstract methods
- Different implementation

Polymorphism: 多形

- A method in the extended class behaves differently from its superclass
- An instance of the extended class can be treated as an instance of its superclass

Limitation in Java inheritance

- Difficulties in multiple inheritance
  - Superclasses have fields or methods with the same name
- Java allows
  - a class inherits only one superclass
- Interfaces as special Superclasses
  - Java classes can inherit multiple interfaces

Interfaces

- Restriction on fields
  - have only static final fields
- Restriction on method
  - Abstract method: no implementation

Using interfaces

- Declare using interface at class definition
- Implement all abstract methods
- Users of the class with the interface need to know only the methods of the interface

Comparable: example of interfaces

- Read API document
  `https://www.oracle.com/jp/java/technologies/documentation.html`
- Understand
  - method
  - return value

Today's tasks

- Working with `example1` package
- Add `Comparable` interface to `Student` class
- Implement `compareTo()` method
- Change `MergeSort` to be compatible with `Comparable` instances

Implement Comparable interface to Student class

- Copy example0/Student.java into example1 package
- Confirm package name in example1/Student.java
- Modify class definition

```
public class Student implements Comparable<Student>
```

- Implement compateTo() method

Modify `MergeSort` compatible with `Comparable`

- Copy `example0/MergeSort.java` into `example1`
- Generalize target class
  - specify the target using class template
  - delete all `Student` class specification

```
public class MergeSort<T extends Comparable<T>>
```

- Do not user `Student` class
  - use `compareTo()` method

Homework

- Write a class for bubble sort compatible `Comparable`
- Use class templates