Object-Oriented Design – Detailed Design

Xin Feng

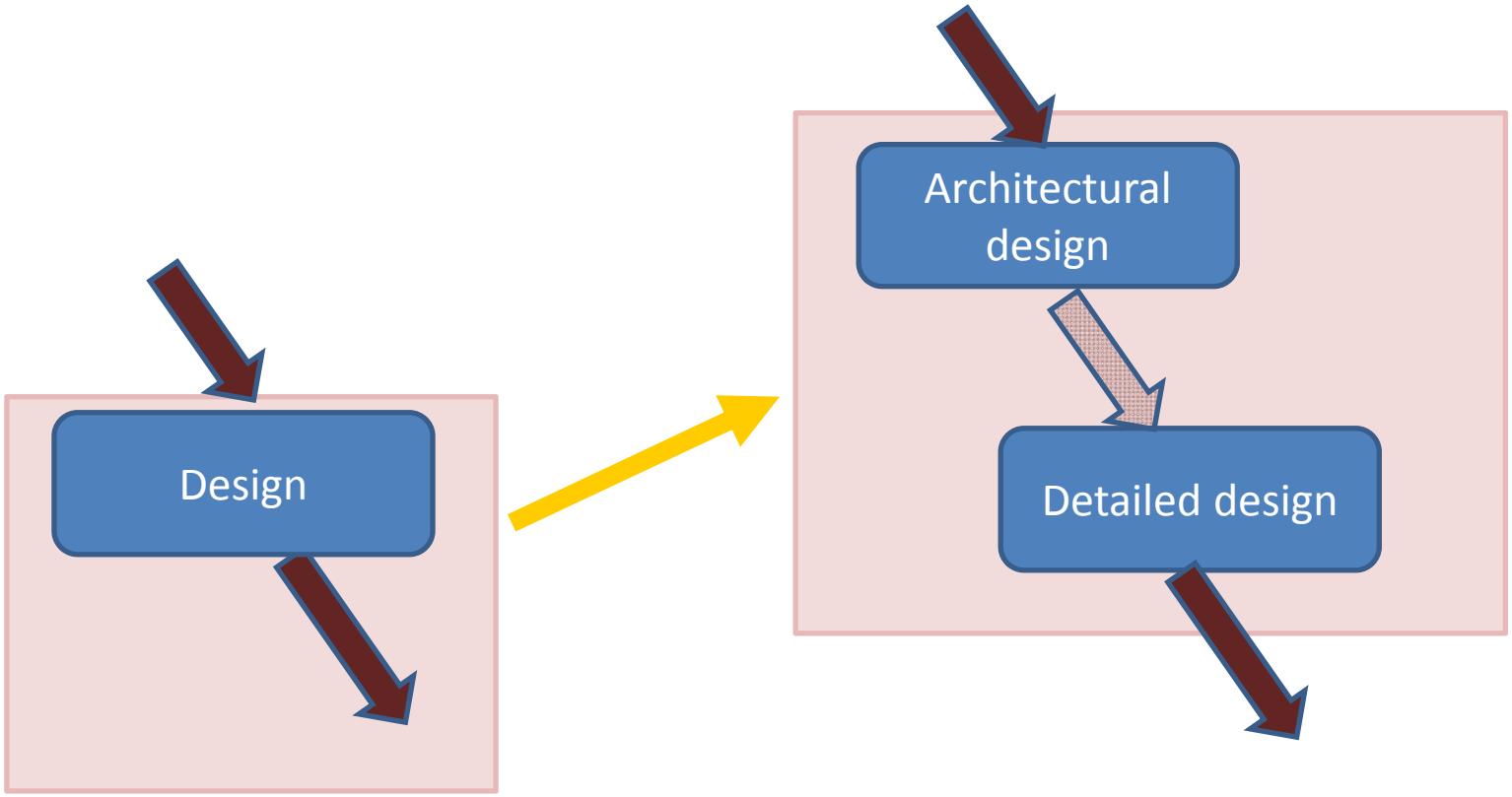
Outline

- Detailed design in OO
- Steps in detailed design

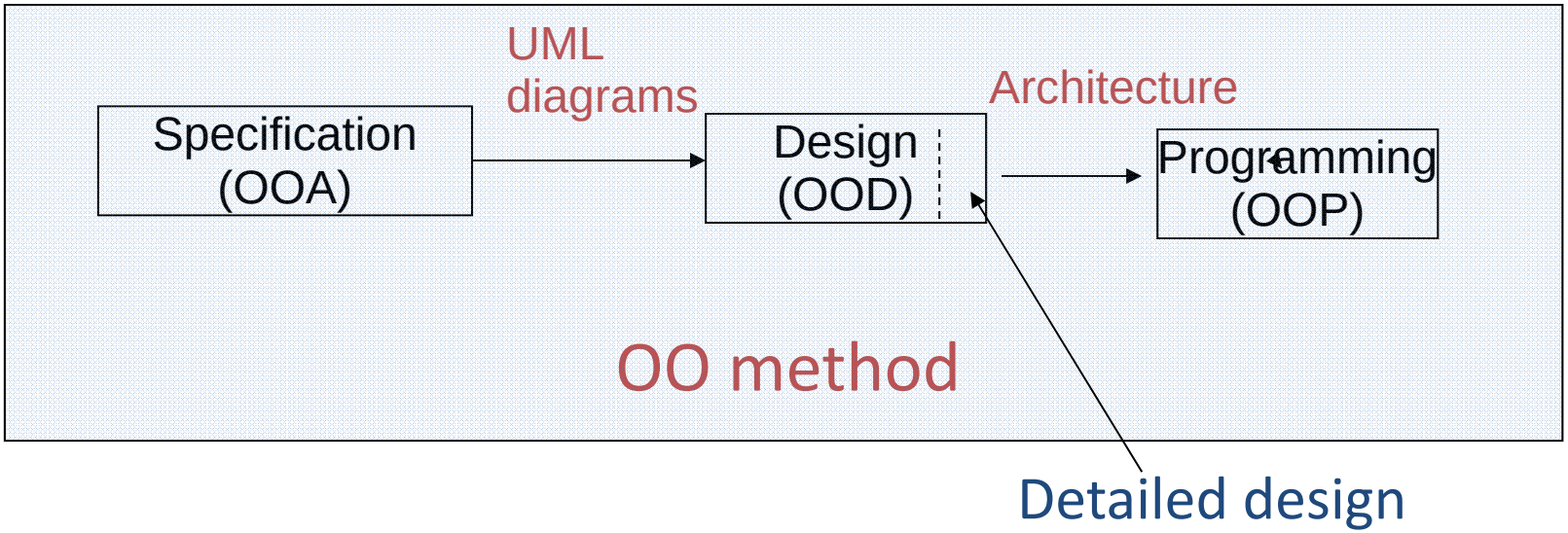
Detailed Design

- Restructure (重新构造) the design for implementation
- Give some detailed （详细）information
- Interface （接口）design

Software Development Life Cycle



Detailed Design

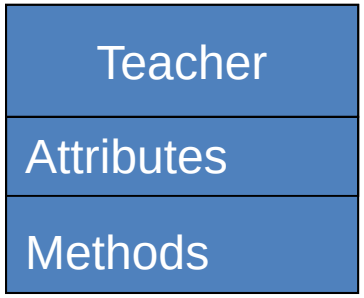


Detailed Design

- OOD is based on
  - UML diagrams
  - MVC UML
- Detailed design
  - Interface design
  - Restructuring (重新构造)

Interface design

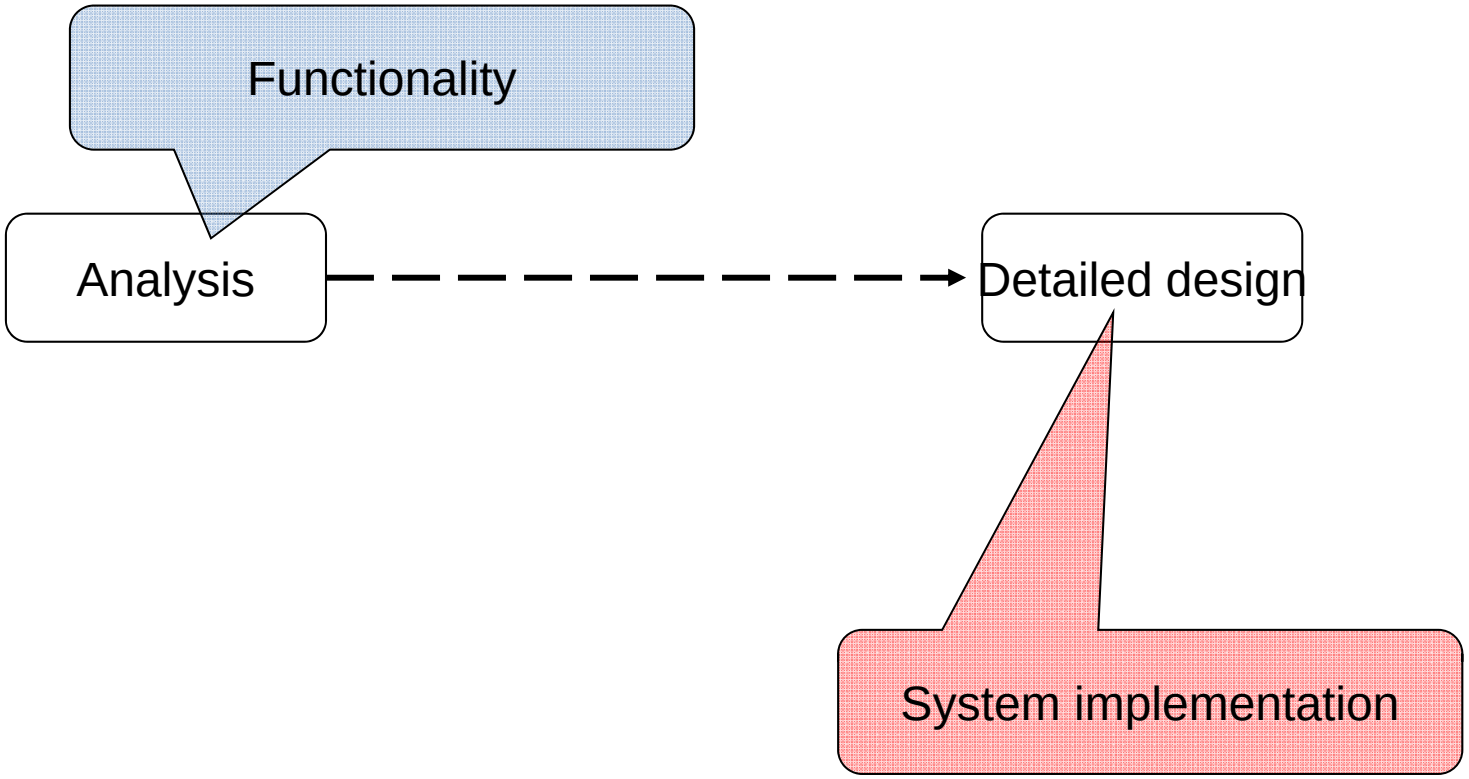
- Identifying （标识）missing （丢失的）attributes and operations
- Specify visibility （可见性）and signature （说明标志）
- Specify pre- and post- conditions （前置条件和后置条件）



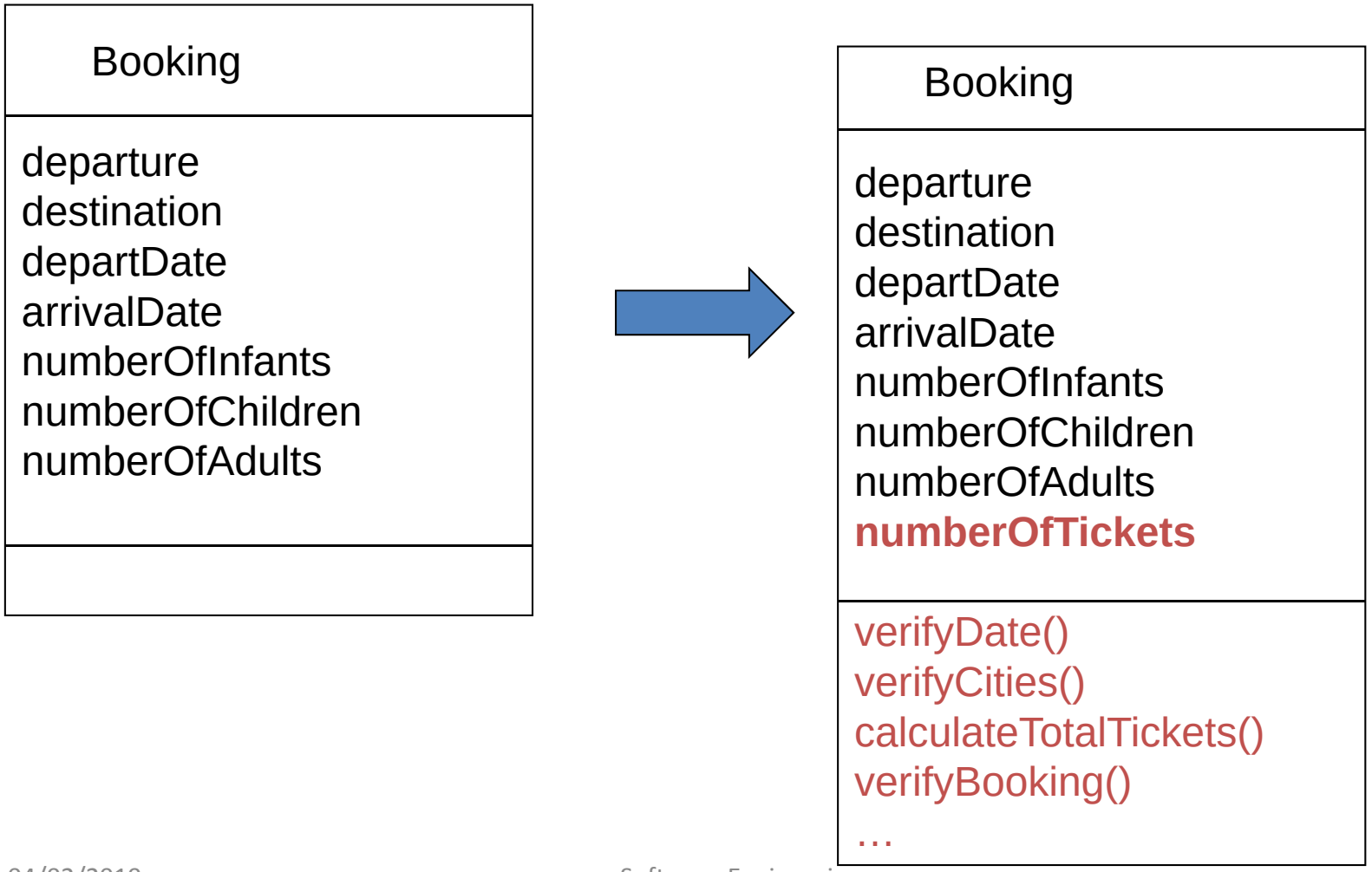
Identifying Missing Attributes and Operations

- For each sub-sytem
  - Examine each service and each participating object
  - Identify missing operations and attributes that are needed to realize the subsystem service
  - Refine （精化） the current object design model and augment （扩大）it with these attributes and operations.

Identifying Missing Attributes and Operations



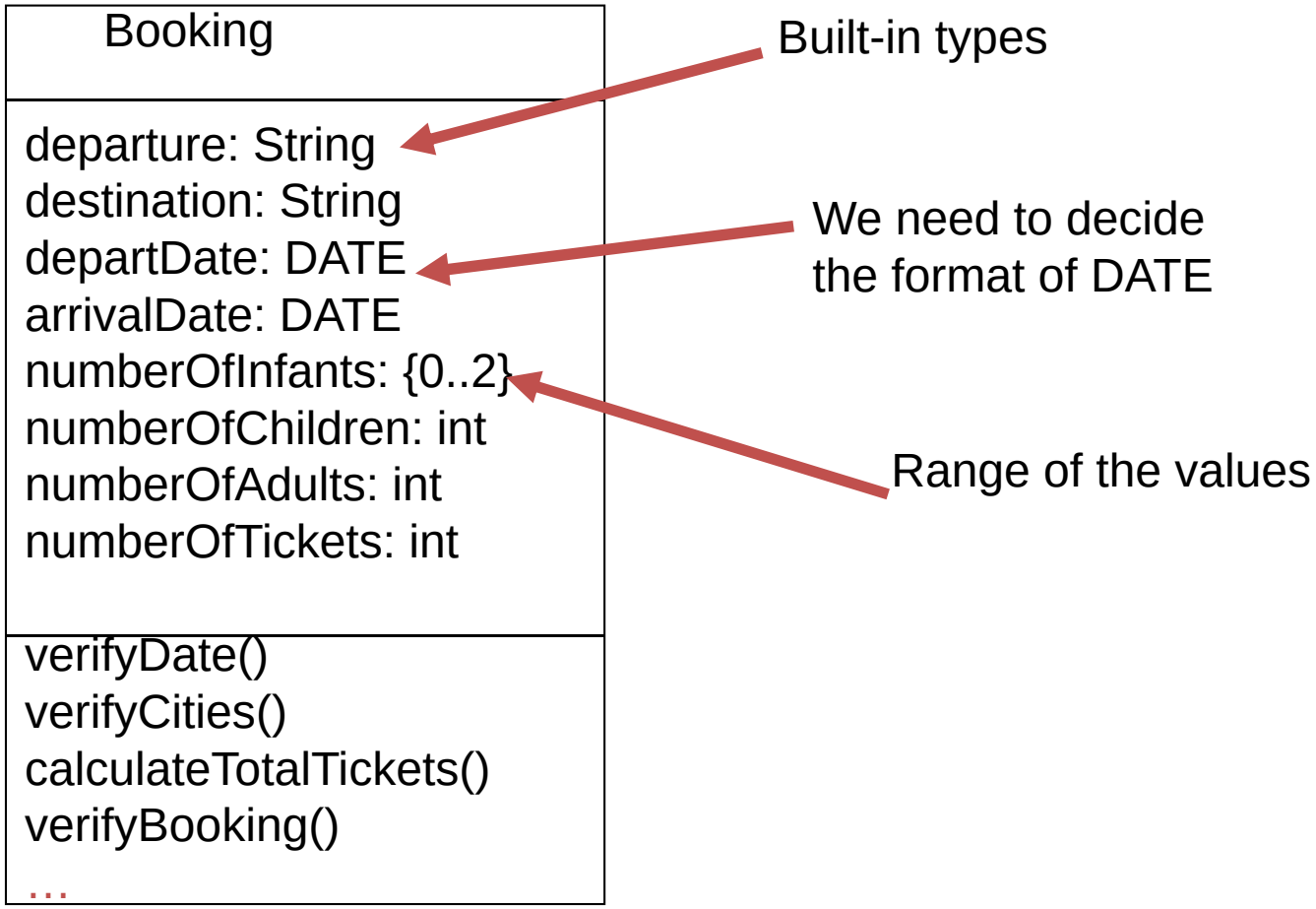
Identifying Missing Attributes and Operations



Specify Visibility (可见性) and Signature （标志说明）

- Types of the attributes
  - Specifying the range of each attribute
  - Map the classes and attributes of the object model to built-in types （内置 的）related to a certain development environment.

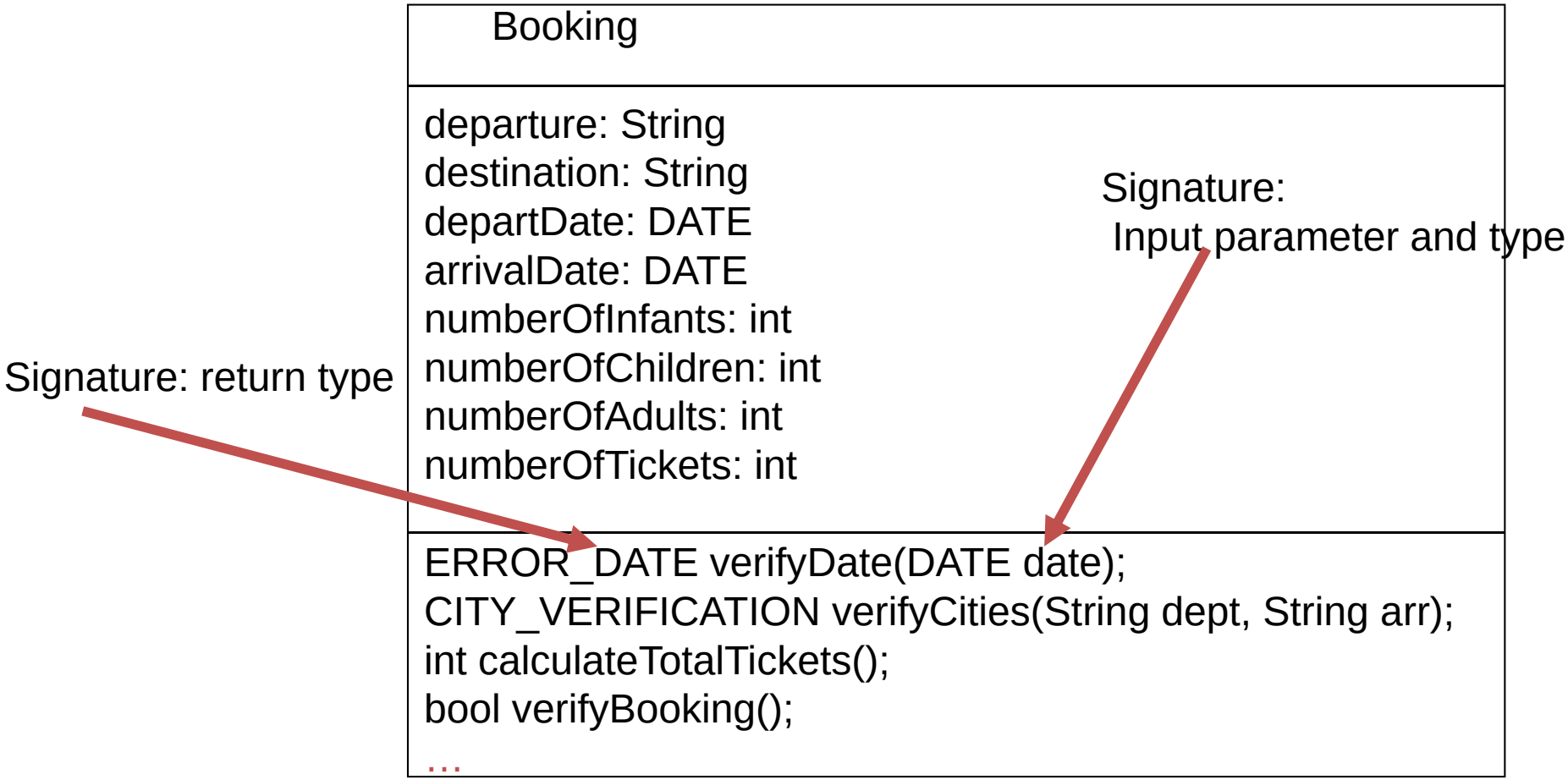
Specify Visibility and Signature



Specify Visibility and Signature

- The signature of an operation includes
  - The return type
  - The operation name
  - The parameters and their types

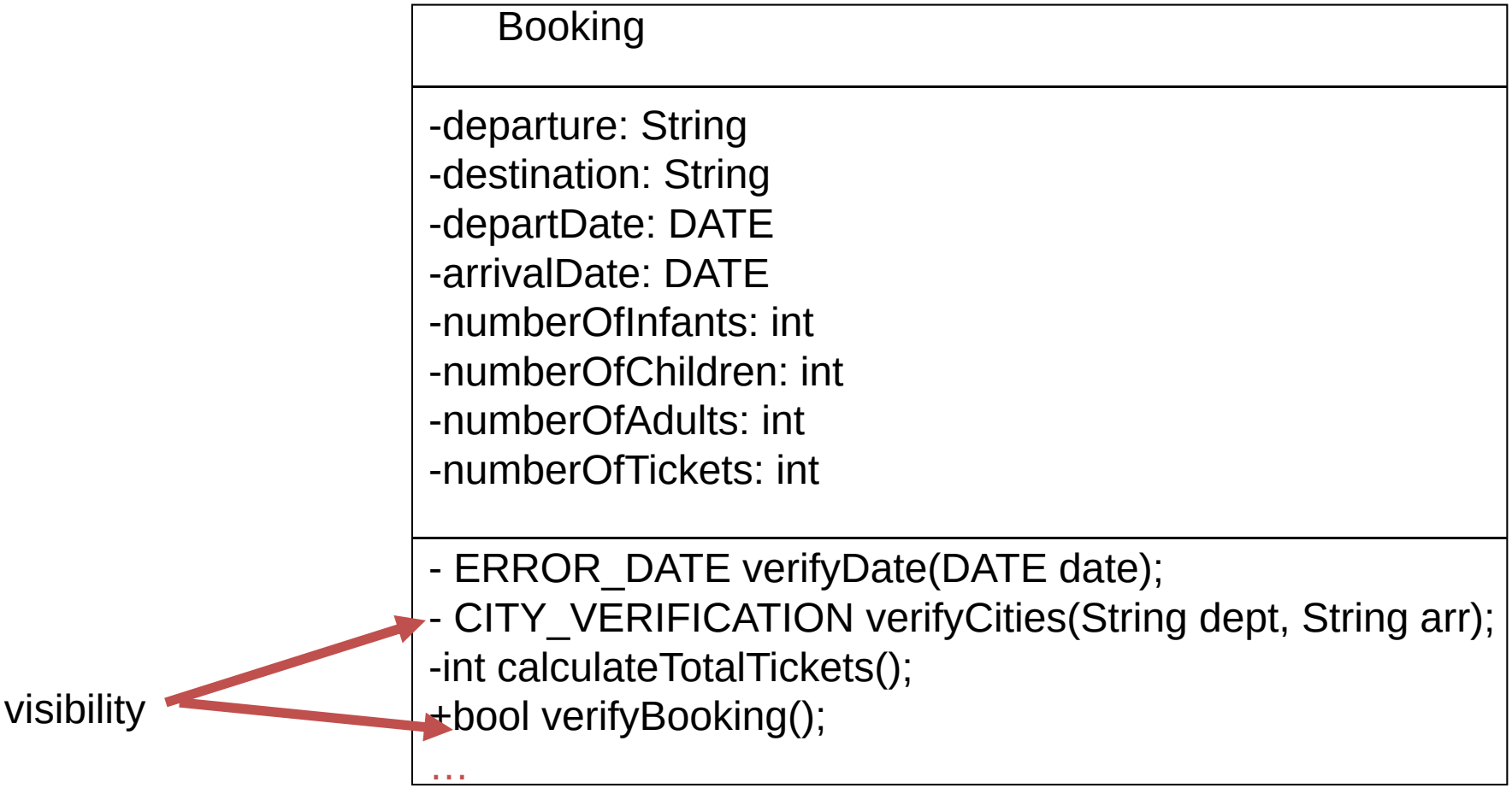
Specify Visibility and Signature



Specify Visibility and Signature

- The visibility of attributes and operations
  - Private (-)
  - Protected (#)
  - Public (+)

Specify Visibility and Signature



Specify pre- and post- conditions

- For each operation, we must specify the conditions that must be satisfied before the operation is invoked and after the return of the results

Specify pre- and post- conditions

CITY_VERIFICATION = {WRONG_CITY_NAME, NO_FLIGHT, OK}

CITY_VERIFICATION verifyCities(String dept, String arr);

Pre-condition: *dept* and *arr* are not null.

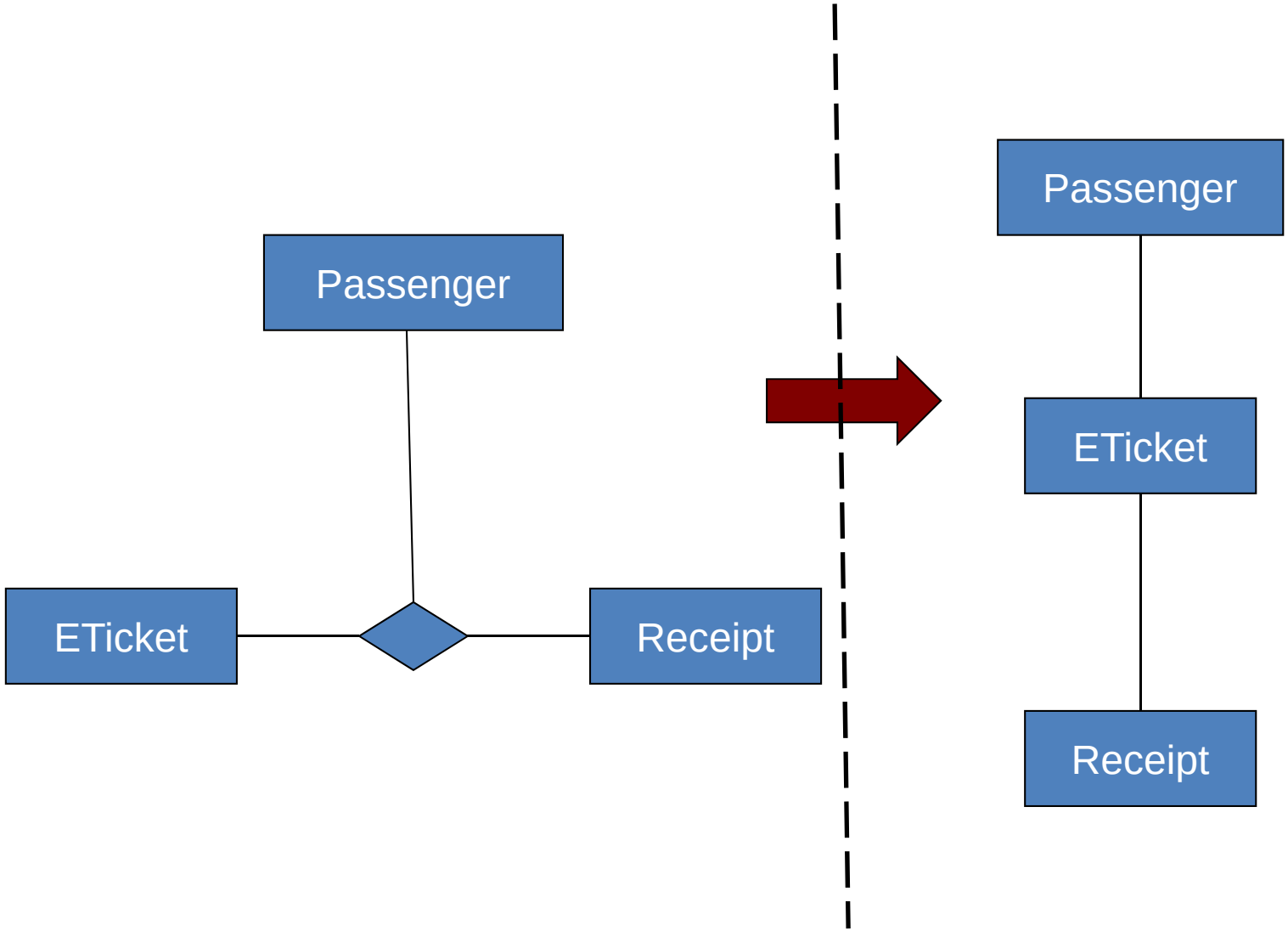
Post-condition: the return value corresponds to the following values

1. if the city name does not exist in the list provided by the company, return WRONG_CITY_NAME;
2. if there are no flights between these two cities, return NO_FLIGHT;
3. otherwsie, return OK.

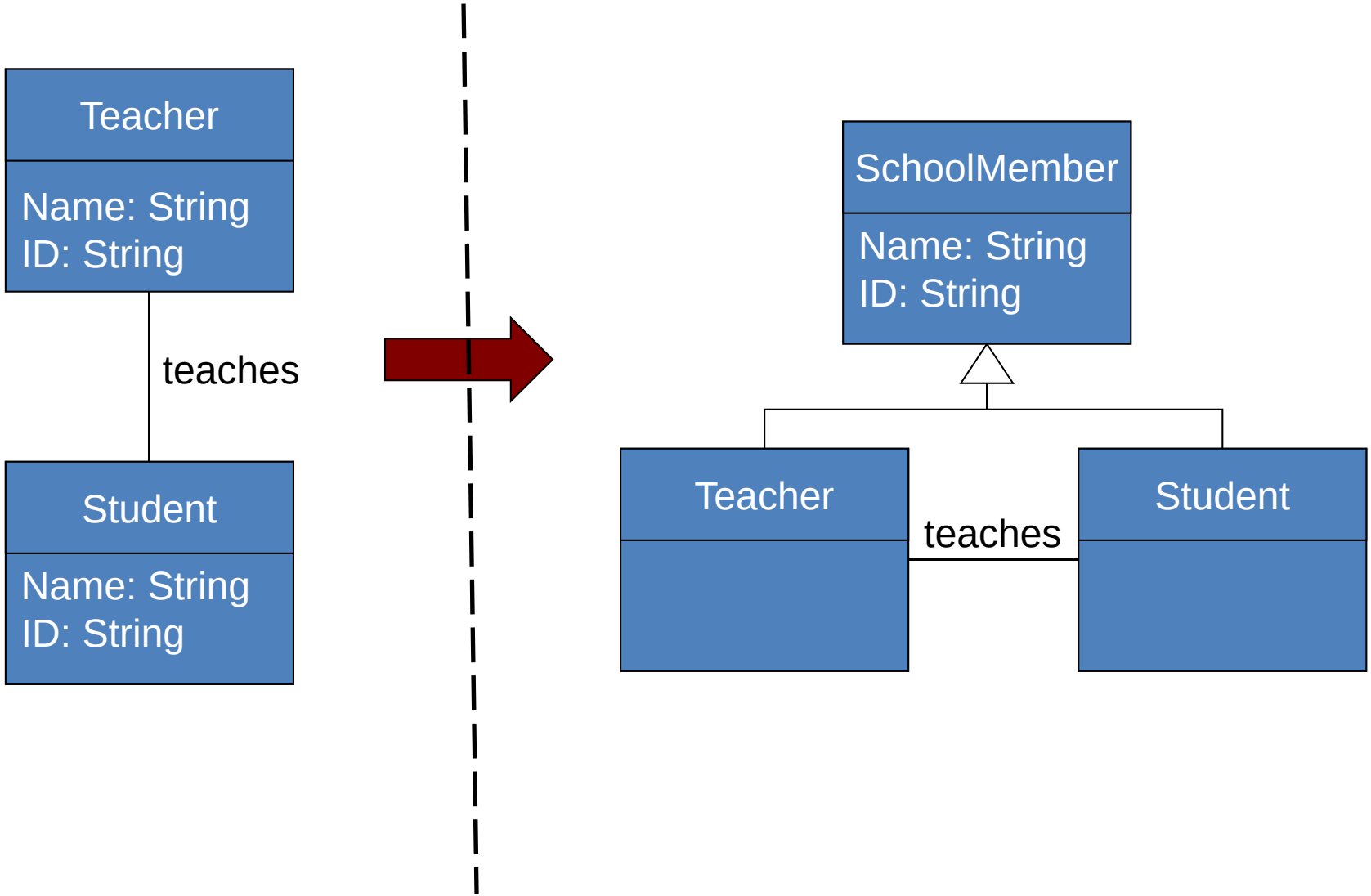
Restructuring

- Change n-ary associations to binary associations
- Increase the inheritance
- Collapse (降级) classes with no significant （明显的） behavior into attributes
- Use qualifier （限定符） to change the one-to-many and many-to-many relationship
- Implement an association class as a class

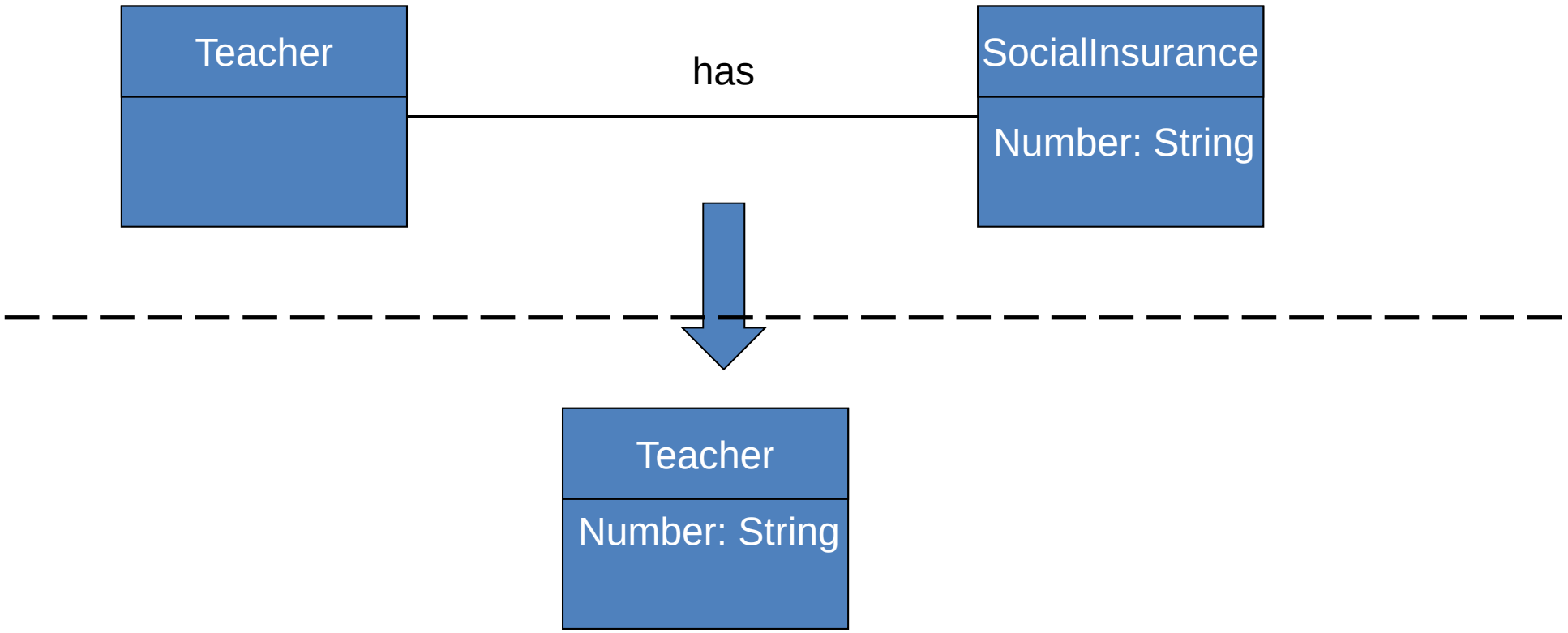
Change N-ary Association to Binary Association



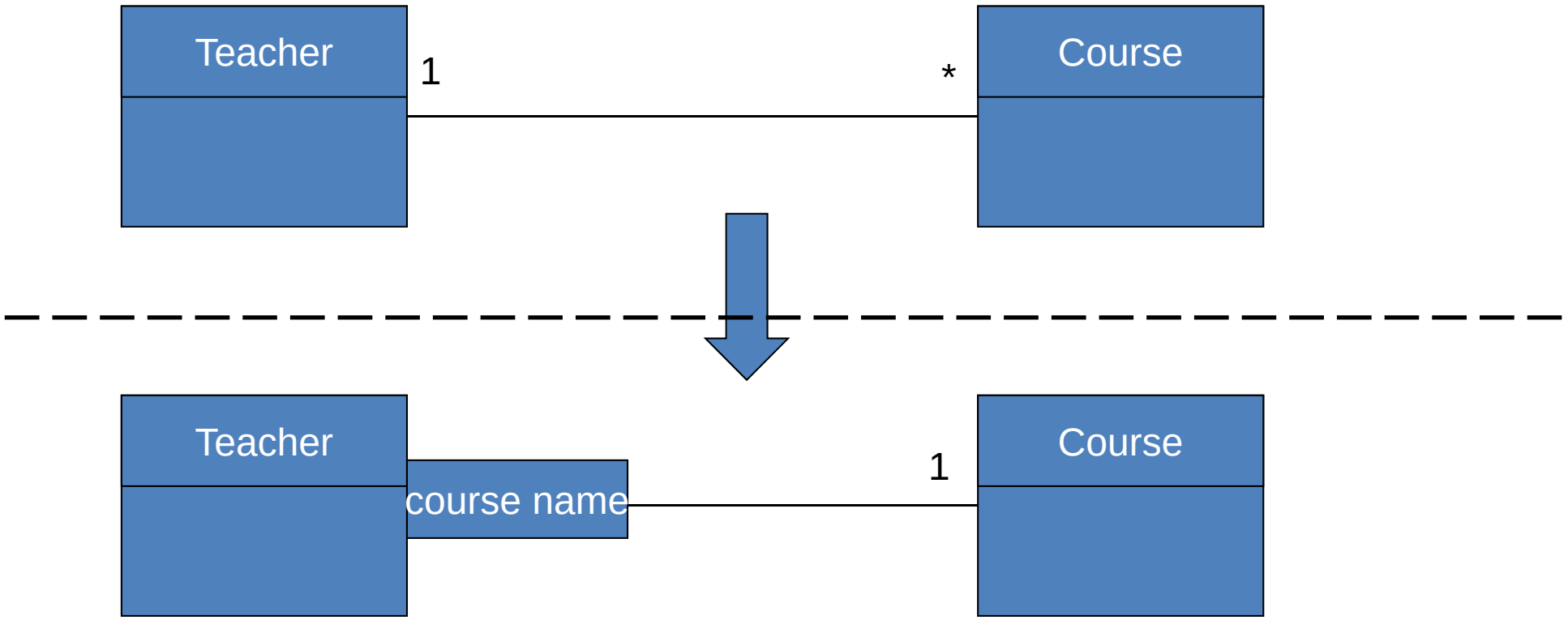
Increase Inheritance



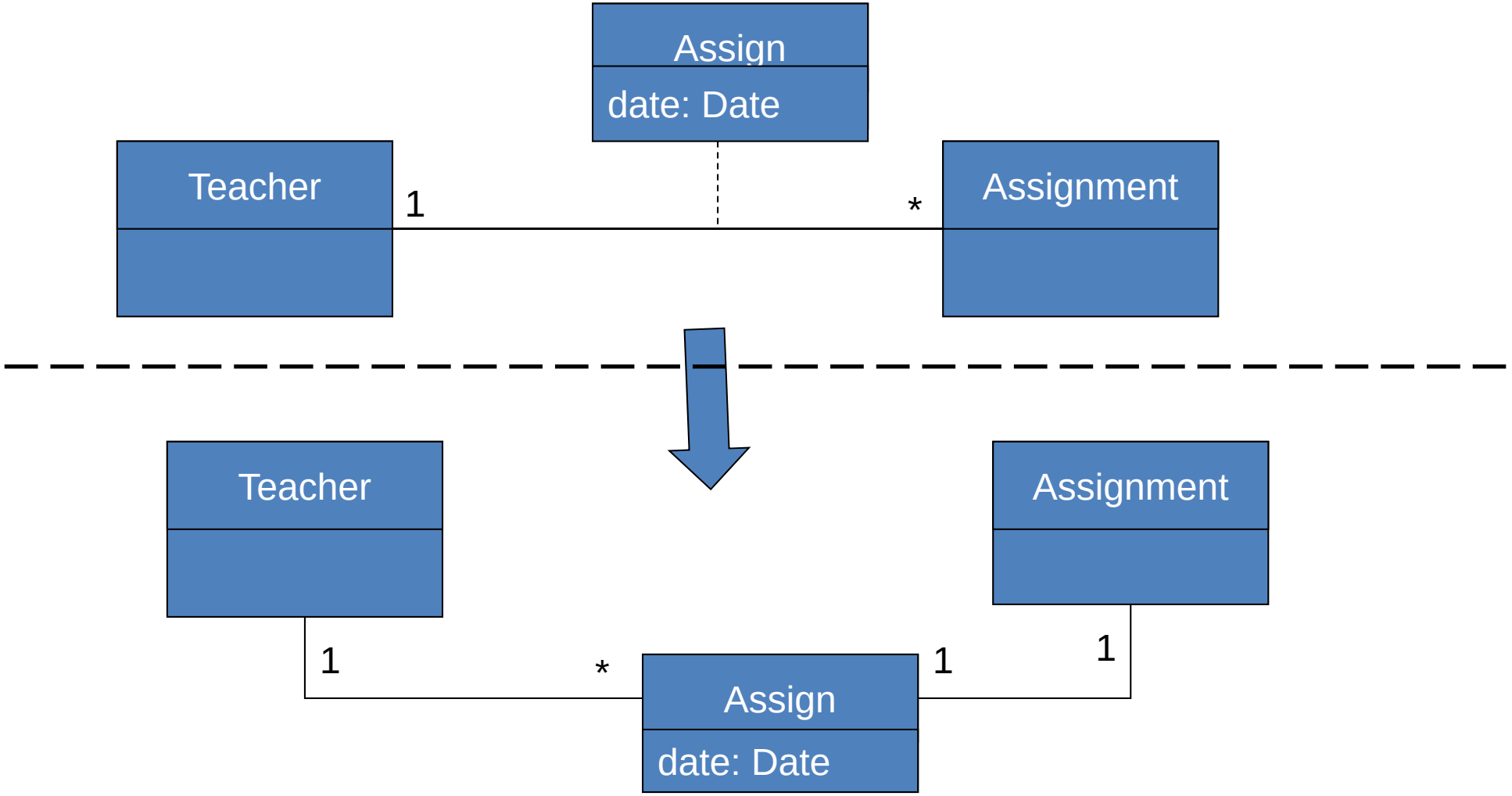
Collapse Classes into Attributes



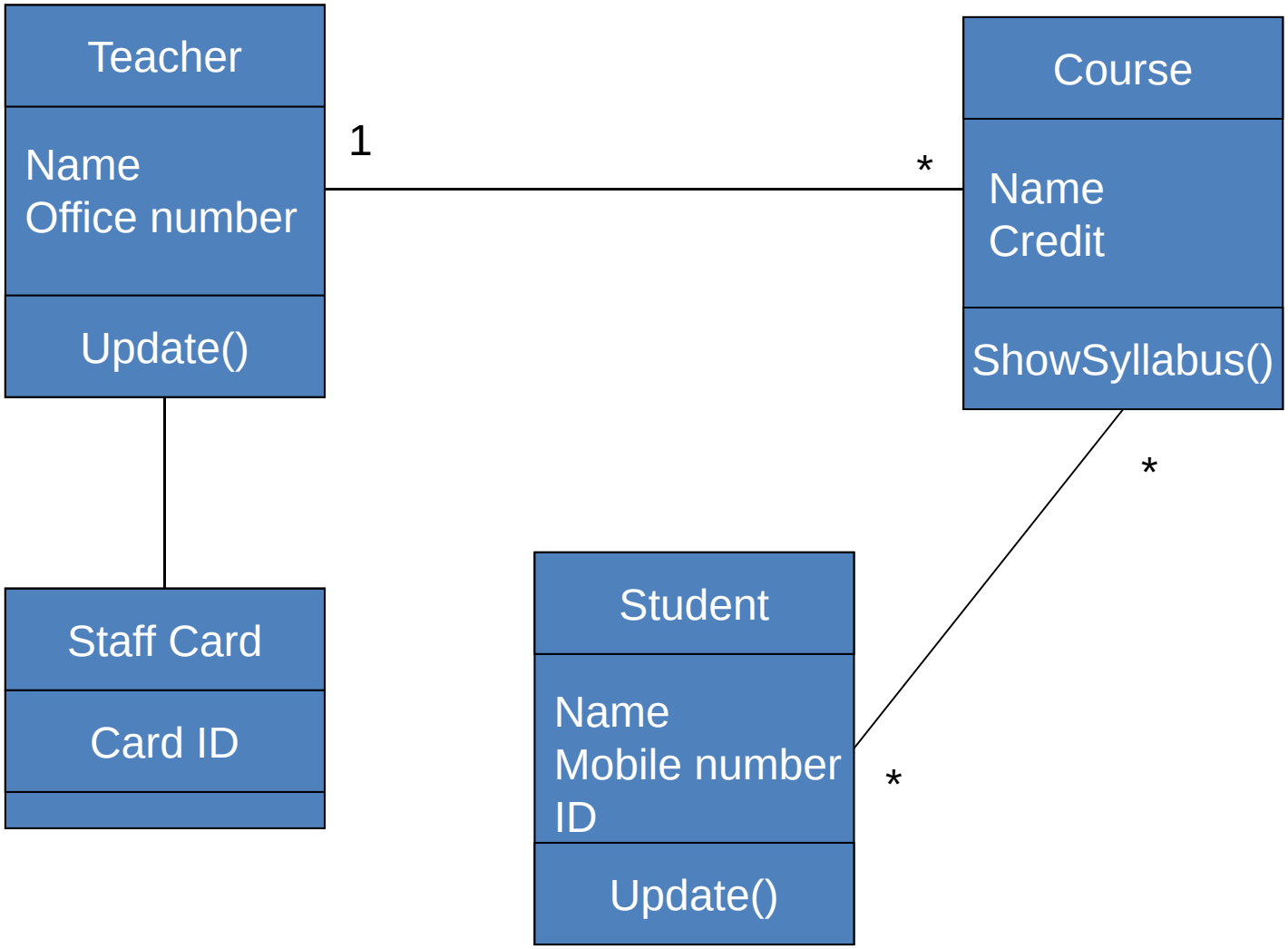
Use Qualifier to Reduce Multiplicity



Implement an Association Class as A class



Class Exercise



Summary

- Interface design is to make the class interface more detailed
- Restructure a class diagram towards implementation